Coling 2010

23rd International Conference on Computational Linguistics

Proceedings of the Conference Volume 2

Chu-Ren Huang and Dan Jurafsky

23 – 27 August 2010
Beijing International Convention Center
Beijing, China

Published by
Tsinghua University Press

Block A, Xue Yan Building, Tsinghua University
Beijing, 100084
China

To order hard copies of this proceedings, contact:

Chinese Information Processing Society of China
No.4, Southern Fourth Street
Haidian District, Beijing, 100190
China
Tel: +86-010-62562916
Fax: +86-010-62562916
cips@iscas.ac.cn

Exploring variation across biomedical subdomains

Tom Lippincott and **Diarmuid Ó Séaghdha** and **Lin Sun** and **Anna Korhonen**
Computer Laboratory
University of Cambridge
{tl318,do242,ls418,alk23}@cam.ac.uk

Abstract

Previous research has demonstrated the importance of handling differences between domains such as "newswire" and "biomedicine" when porting NLP systems from one domain to another. In this paper we identify the related issue of *subdomain variation*, i.e., differences between subsets of a domain that might be expected to behave homogeneously. Using a large corpus of research articles, we explore how subdomains of biomedicine vary across a variety of linguistic dimensions and discover that there is rich variation. We conclude that an awareness of such variation is necessary when deploying NLP systems for use in single or multiple subdomains.

1 Introduction

One of the most noticeable trends in the past decade of Natural Language Processing (NLP) research has been the deployment of language processing technology to meet the information retrieval and extraction needs of scientists in other disciplines. This meeting of fields has proven mutually beneficial: scientists increasingly rely on automated tools to help them cope with the exponentially expanding body of publications in their field, while NLP researchers have been spurred to address new conceptual problems in theirs. Among the fundamental advances from the NLP perspective has been the realisation that tools which perform well on textual data from one source may fail to do so on another unless they are tailored to the new source in some way. This has led to significant interest in the idea of contrasting *domains* and the concomitant problem of *domain adaptation*, as well as the production of manually annotated domain-specific corpora.[1]

One definition of *domain variation* associates it with differences in the underlying probability distributions from which different sets of data are drawn (Daumé III and Marcu, 2006). The concept also mirrors the notion of variation across thematic subjects and the corpus-linguistic notions of *register* and *genre* (Biber, 1988). In addition to the differences in vocabulary that one would expect to observe, domains can vary in many linguistic variables that affect NLP systems. The scientific domain which has received the most attention (and is the focus of this paper) is the biomedical domain. Notable examples of corpus construction projects for the biomedical domain are PennBioIE (Kulick et al., 2004) and GENIA (Kim et al., 2003). These corpora have been used to develop systems for a range of processing tasks, from entity recognition (Jin et al., 2006) to parsing (Hara et al., 2005) to coreference resolution (Nguyen and Kim, 2008).

An implicit assumption in much previous work on biomedical NLP has been that particular subdomains of biomedical literature – typically molecular biology – can be used as a model of biomedical language in general. For example, GENIA consists of abstracts dealing with a specific set of subjects in molecular biology, while PennBioIE covers abstracts in two specialised domains, cancer genomics and the behaviour of a particular class of enzymes. This assumption of representativeness is understandable because linguistic annotation is labour-intensive and it may not be worthwhile to produce annotated corpora for multiple subdomains within a single discipline if there is lit-

[1] A workshop dedicated to domain adaptation is collocated with ACL 2010.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 689–697,
Beijing, August 2010

tle task-relevant variation across those subdomains. However, such conclusions should not be made before studying the actual degree of difference between the subdomains of interest.

One of the principal goals of this paper is to map how the concept of "biomedical language", often construed as a monolithic entity, is composed of diverse patterns of behaviour at more fine-grained topical levels. Hence we study linguistic variation in a broad biomedical corpus of abstracts and full papers, the PMC Open Access Subset.[2] We select a range of lexical and structural phenomena for quantitative investigation. The results indicate that common subdomains for resource development are not representative of biomedical text in general and furthermore that different linguistic features often partition the subdomains in quite different ways.

2 Related Work

A number of researchers have explored the differences between non-technical and scientific language. Biber and Gray (2010) describe two distinctive syntactic characteristics of academic writing which set it apart from general English. Firstly, in academic writing additional information is most commonly integrated by pre- and post-modification of phrases rather than by the addition of extra clauses. Secondly, academic writing places greater demands on the reader by omitting non-essential information, through the frequent use of passivisation, nominalisation and noun compounding. Biber and Gray also show that these tendencies towards "less elaborate and less explicit" language have become more pronounced in recent history.

We now turn to corpus studies that focus on biomedical writing. Verspoor et al. (2009) use measurements of lexical and structural variation to demonstrate that Open Access and subscription-based journal articles in a specific domain (mouse genomics) are sufficiently similar that research on the former can be taken as representative of the latter. While their primary goal is different from ours and they do not consider variation across multiple domains, they do compare their mouse genomics corpus with small reference corpora drawn from newswire and general biomedical sources. This analysis unsurprisingly finds differences between the domain and newswire corpora across many linguistic dimensions; more interestingly for our purposes, the comparison of domain text to the broader biomedical superdomain shows a more complex picture with similarities in some aspects (e.g., passivisation and negation) and dissimilarities in others (e.g., sentence length, semantic features).

Friedman et al. (2002) document the "sublanguages" associated with two biomedical domains: clinical reports and molecular biology articles. They set out restricted ontologies and frequent co-occurrence templates for the two domains and discuss the similarities and differences between them, but they do not perform any quantitative analysis.

Other researchers have focused on specific phenomena, rather than cataloguing a broad scope of variation. Cohen et al. (2008) carry out a detailed analysis of argument realisation with respect to verbs and nominalisations, using the GENIA and PennBioIE corpora. Nguyen and Kim (2008) compare the behaviour of anaphoric pronouns in newswire and biomedical corpora; they improve the performance of a pronoun resolver by incorporating their observations, thus demonstrating the importance of capturing domain-specific phenomena. Nguyen and Kim's findings are discussed in more detail in Section 5.4 below.

3 Subdomains in the OpenPMC Corpus

The Open Access Subset of PubMed (OpenPMC) is the largest publicly available corpus of full-text articles in the biomedical domain. OpenPMC is comprised of 169,338 articles drawn from 1233 medical journals, totalling approximately 400 million words. The NIH maintains a one-to-many mapping from journals to 122 subject areas (NIH, 2009b). This covers about 400 of the OpenPMC journals, but these account for over 70% of the database by byte size and word count. Journals are assigned up to five subject areas with the majority assigned one (69%) or two (26%) subjects. In this paper we adopt the OpenPMC subject areas (e.g. "Pulmonary Medicine", "Genetics", "Psychiatry") as the basis for subdomain comparison.

[2] http://www.ncbi.nlm.nih.gov/pmc/about/openftlist.html

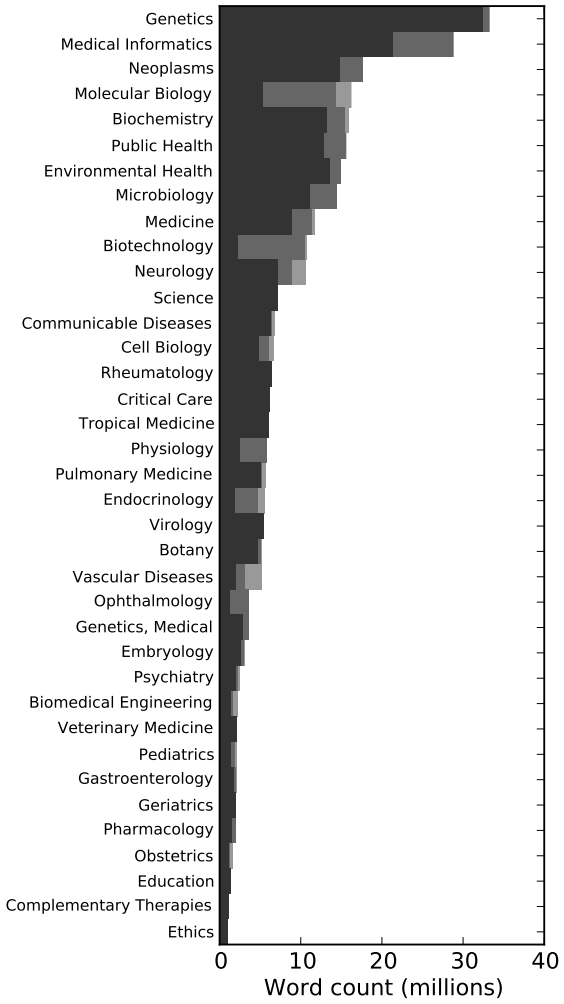


Figure 1: OpenPMC word count by subdomain, dark colouring indicates data assigned single subdomain, each lighter shade indicates an additional overlapping subdomain

4 Methodology

4.1 Data selection and preprocessing

An important initial question was how to treat data with multiple classifications: we only consider journals assigned a single subdomain, to avoid the added complexity of interactions in data from overlapping subdomains. To ensure sufficient data for comparing a variety of linguistic features, we discard the subdomains with less than one million words meeting the single-subdomain criterion. After review, we also drop the "Biology" subdomain, which appears to function as a catch-all for many loosely related areas. Figure 1 shows the distribution of data across the subjects we use, by word-count, with lighter-coloured areas representing data that is assigned multiple subjects. These subjects provide a convenient starting point for dividing the corpus into subdomains (hereafter, "subdomain" will be used rather than "subject"). We also add a reference subdomain, "Newswire", composed of a 6 million word random sample from the English Gigaword corpus (Graff et al., 2005). The final data set has a total of 39 subdomains.

Articles in the OpenPMC corpus are formatted according to a standard XML tag set (NIH, 2009a). We first convert each article to plain text, ignoring "non-content" elements such as tables and formulas, and split the result into sentences, aggregating the results by subdomain.

4.2 Feature extraction

We investigate subdomain variation in our corpus across a range of lexical, syntactic, sentential and discourse features. The corpus is lemmatised, tagged and parsed using the C&C pipeline (Curran et al., 2007) with the adapted part-of-speech and lexical category tagging models produced by Rimell and Clark (2009) for biomedical parsing.

From this output we count occurrences of noun, verb, adjective and adverb lemmas, part-of-speech (POS) tags, grammatical relations (GRs), chunks, and lexical categories. The lemma features are Zipfian-distributed items from an open class, so we have experimented with filtering low-frequency items at various thresholds to reduce noise and improve processing speed. The other feature sets can be viewed as closed classes, where filtering is unnecessary.

Since verbs are central to the meaning and structure of sentences, we consider their special behavior by constructing features for each verb's distribution over other grammatical properties. Several grammatical properties are captured by pairing each verb with its POS (indicating e.g. tense, such as present, past, and present participle). Voice is determined from additional annotation output by the C&C parser. Table 1 shows the POS-distribution for the verb "restrict", in two subdomains from the corpus. Finally, we record distributions over verb subcategorization frames (SCFs) taken by each verb, and over the GRs it participates in.

Subdomain	VB	VBG	VBN	VBP	VBZ
Medical Informatics	.35	.29	.06	.09	.21
Cell Biology	.14	.43	.05	.10	.29

Table 1: Distribution over POS tags for verb "restrict", in two subdomains

SCFs were extracted using a system of Preiss et al. (2007).

To facilitate a more robust and interpretable analysis of vocabulary differences, we estimate a "topic model" of the corpus with Latent Dirichlet Analysis (Blei et al., 2003) using the MALLET toolkit.[3] As preprocessing we divide the corpus into articles, removing stopwords and words shorter than 3 characters. The Gibbs sampling procedure is parameterised to induce 100 topics, each giving a coherent cluster of related words learned from the data, and to run for 1000 iterations. We collate the predicted distribution over topics for each article in a subdomain, weighted by article wordcount, to produce a topic distribution for the subdomain.

4.3 Measurements of divergence

Our goal is to illustrate the presence or absence of differences between the feature sets, and to do so we calculated the Jensen-Shannon divergence and the Pearson correlation. Jensen-Shannon divergence is a finite symmetric measurement of the divergence between probability distributions, while Pearson correlation quantifies the linear relationship between two real-valued samples.

The count-features are weighted, for a given subdomain, by the feature's log-likelihood between the subdomain's data and the rest of the corpus. Log-likelihood has been shown to perform well when comparing counts of potentially low-frequency features (Rayson and Garside, 2000) such as found in Zipfian-distributed data. This serves to place more weight in the comparison on items that are distinctive of the subdomain with respect to the entire corpus.

While the count-features are treated as a single distribution for the purposes of JSD, the verbwise-features are composed of many distributions, one for each verb lemma. Our approach is to combine the JSD of the verbs, weighted by the log-likelihood of the verb lemma between the two subdomains in question, and normalize the distances to the interval [0, 1]. Using the lemma's log-likelihood assumes that, when a verb's distribution behaves differently in a subdomain, its frequency changes as well.

We present the results as dendrograms and heat maps. Dendrograms are tree structures that illustrate the results of hierarchical clustering. We perform hierarchical clustering on the inter-subdomain divergences for each set of features. The algorithm begins with each instance (in our case, subdomains) as a singleton cluster, and repeatedly joins the two most similar clusters until all the data is clustered together. The order of these merges is recorded as a tree structure that can be visualized as a dendrogram in which the length of a branch represents the distance between its child nodes. Similarity between clusters is calculated using average distance between all members, known as "average linking".

Heat maps show the pairwise calculation of a metric in a grid of squares, where square (x, y) is shaded according to the value of $metric(sub_x, sub_y)$. For our measurements of JSD, black represents 0 (i.e. identical distributions) and white represents the metric's theoretical maximum of 1. We also inscribe the actual value inside each square. Dendrograms are tree structures that illustrate the hierarchical clustering procedure described above. The dendrograms present all 39 subdomains, while for readability the heatmaps present 12 subdomains selected for representativeness.

5 Results

Different thresholds for filtering low-frequency terms had little effect on the divergence measures, and served mainly to improve processing time. We therefore report results using a cutoff of 150 occurrences (over the entire 234 million word data set) and log-likelihood weights. The results of Pearson correlation and JSD show similar trends, and due to its specific design for comparing distributions we only report the latter.

[3] http://mallet.cs.umass.edu

5.1 Vocabulary and lexical features

Differences in vocabulary are what first comes to mind when describing subdomains. Word features are fundamental components for systems such as POS taggers and lexicalised parsers; one therefore expects that these systems will be affected by variation in lexical distributions. Figure 2a uses JSD calculated on each subdomain's distribution over 100 LDA-induced topics to compare vocabulary distributions. Subdomains related to molecular biology (Genetics, Molecular Biology) show the smallest divergences, an interesting fact since these are heavily used in building resources for BioNLP. The dendrogram shows a rough division into "public policy", "patient-centric", "applied" and "microscopic" subdomains, with the distance between unrelated subdomains such as Biochemistry and Pediatrics almost as large as their respective differences from Newswire.

We omit figures for variation over noun, verb and adjective lemmas due to space restrictions; in general, these correlate with the variation in LDA topics though there are some differences. Figure 2b shows JSD calculated on distributions over adverb lemmas. Part of the variation is due to characteristic markers of scientific argument ("therefore", "significantly", "statistically"). A more interesting factor is the coining of domain-specific adverbs, an example of the tendency in scientific text to use complex lexical items and premodifiers rather than additional clauses. This also has the effect of moving subdomain-specific objects and processes from verbs and nouns to adverbs. This behavior seems non-continuous, in that subdomains either make heavy, or almost no, use of it: for example, Pediatrics has no subdomain-specific items among the its ten top adverbs by log-likelihood, while Neoplasms has "histologically", "immunohistochemically" and "subcutaneously". These information-dense terms could prove useful for tasks like automatic curation of subdomain vocabularies, where they imply relationships between their components, the items they modify, etc.

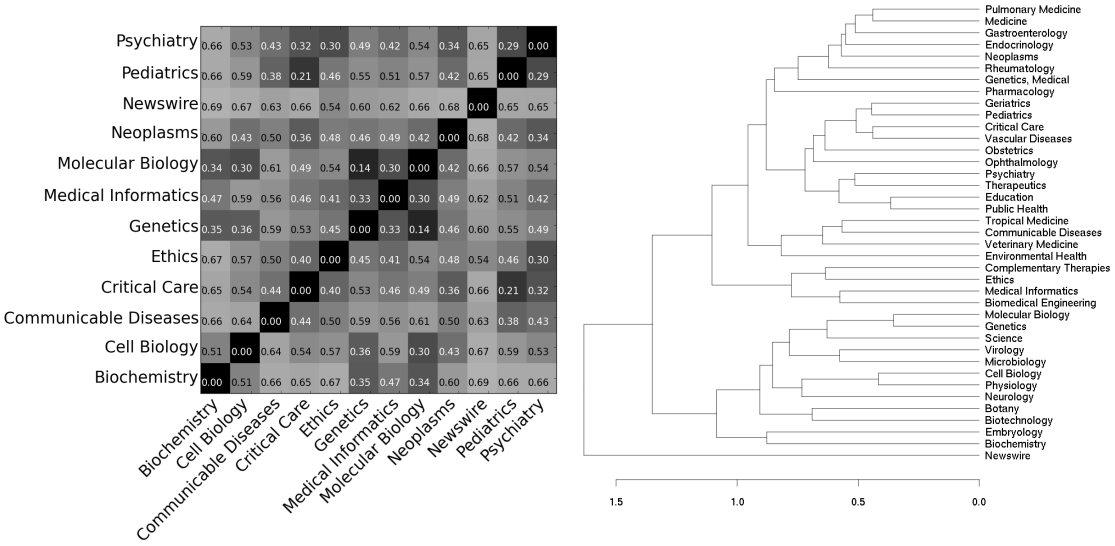
5.2 Verb distributional behavior

Modelling verb behavior is important for both syntactic (Collins, 2003) and semantic (Korhonen et al., 2008) processing, and subdomains are known to conscript verbs into specific roles that change the distributions of their syntactic properties (Roland and Jurafsky, 1998). The four properties we considered verbs' distributions over (SCF, POS, GR and voice) produced similar inter-subdomain JSD values. Figure 2c demonstrates how verbs differ between subdomains with respect to SCFs. For example, while the Pediatrics subdomain uses the verb "govern" in a single SCF among its 12 possibilities, the Genetics subdomain distributes its usage over 7 of them. Two subdomains may both use "restrict" with high frequency (e.g. Molecular Biology and Ethics), but with different frequency distributions over SCFs.

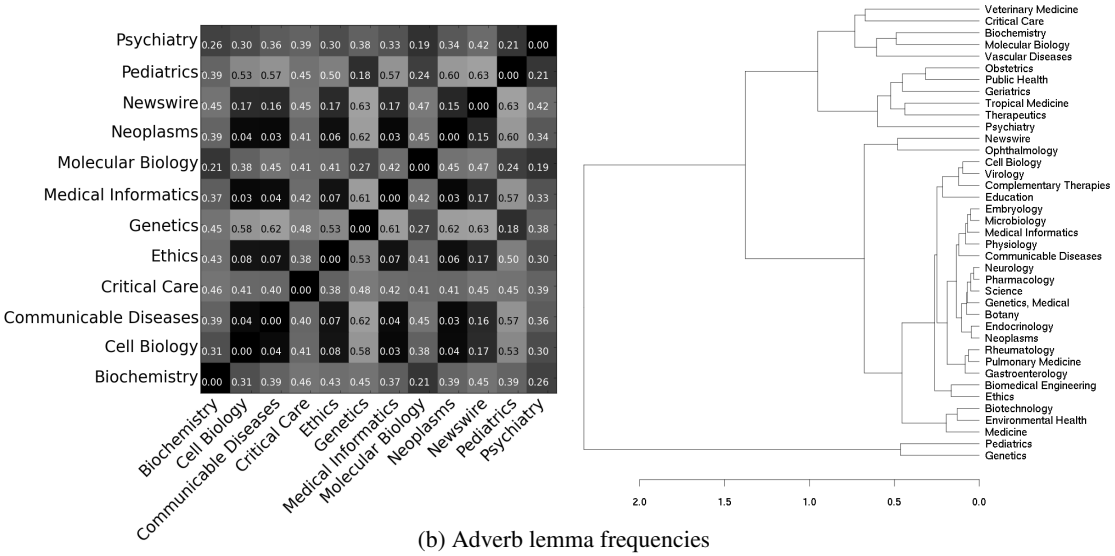
5.3 Syntax

It is difficult to measure syntactic complexity accurately without access to a hand-annotated treebank, but it is well-known that sentence length correlates strongly with processing difficulty (Collins, 1996). The first column of Table 2 gives average sentence lengths (excluding punctuation and "sentences" of fewer than three words) for selected domains. All standard errors are < 0.1. It is clear that all biomedical subdomains typically use longer sentences than newswire, though there is also variation within biomedicine, from an average length of 27 words in Molecular Biology to 24.5 words in Pediatrics.

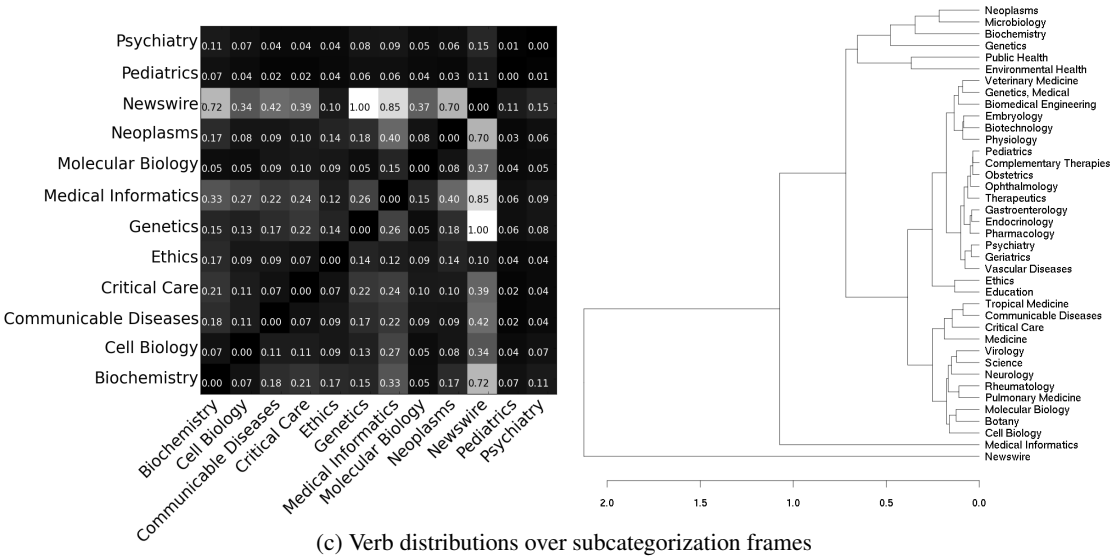
"Packaging" information in complex pre- and/or post-modified noun phrases is a characteristic feature of academic writing (Biber and Gray, 2010). This increases the information density of a sentence but brings with it syntactic and semantic ambiguities. For example, the difficulty of resolving the internal structure of noun-noun compounds and strings of prepositional phrases has been the focus of ongoing research in NLP; these phenomena have also been identified as significant challenges in biomedical language processing (Rosario and Hearst, 2001; Schuman and Bergler, 2006). The second and third columns of Table 2 present average lengths for full noun phrases, defined as every word dominated by a head noun in the grammatical relation graph for a sentence, and for base nominals, defined as nouns plus premodifying adjectives and nouns only. All standard errors are ≤ 0.01. Newswire text uses the simplest noun



(a) LDA-induced distribution over topics



(b) Adverb lemma frequencies



(c) Verb distributions over subcategorization frames

Figure 2: Subdomain variation plotted as heat maps and dendrograms

Sentence length		Full NP length		Base nominal length	
Mol. Biology	27.0	Biochemistry	4.03	Biochemistry	1.85
Genetics	26.6	Genetics	3.90	Neoplasms	1.85
Cell Biology	26.3	Critical Care	3.86	Mol. Biology	1.84
Ethics	26.2	Neoplasms	3.85	Genetics	1.83
PMC Average	25.9	PMC Average	3.85	PMC Average	1.80
Biochemistry	25.8	Pediatrics	3.84	Cell Biology	1.80
Neoplasms	25.5	Med. Informatics	3.84	Critical Care	1.80
Psychiatry	25.3	Comm. Diseases	3.81	Med. Informatics	1.78
Critical Care	25.0	Therapeutics	3.80	Comm. Diseases	1.78
Therapeutics	24.9	Mol. Biology	3.79	Therapeutics	1.75
Comm. Diseases	24.9	Psychiatry	3.77	Psychiatry	1.75
Med. Informatics	24.6	Ethics	3.69	Pediatrics	1.73
Pediatrics	24.6	Cell Biology	3.55	Ethics	1.65
Newswire	19.1	Newswire	3.18	Newswire	1.60

Table 2: Average sentence, NP and base nominal lengths across domains

phrase structures; there is notable variation across PMC domains. Full NP and base nominal lengths do not always correlate; for example, Cell Biology uses relatively long base NPs (nominalisations and multitoken names in particular) but relatively simple full NP structures.

5.4 Coreference

Resolving coreferential terms is a crucial and challenging task when extracting information from texts in any domain. Nguyen and Kim (2008) compare the use of pronouns in the newswire and biomedical domains, using the GENIA corpus as representative of the latter. Among the differences observed between the domains were the absence of any personal pronouns other than third-person neuter pronouns in the GENIA corpus, and a greater proportion of demonstrative pronouns in GENIA than in the ACE or MUC newswire corpora. Corroborating the importance of domain modelling, Nguyen and Kim demonstrate that tailoring a pronoun resolution system to specific properties of the biomedical domain improves performance.

As our corpus is not annotated for coreference we restrict our attention to types that are reliably coreferential: masculine/feminine personal pronouns (*he*, *she* and case variations), neuter personal pronouns (*they*, *it* and variations) and definite NPs with demonstrative determiners such as *this* and *that*. To filter out pleonastic pronouns we used a combination of the C+C parser's pleonasm tag and heuristics based on Lappin and Leass (1994). To filter out the most common class of non-anaphoric demonstrative NPs we simply discarded any matching the pattern *this...paper|study|article*.

Table 3 presents statistics for selected types of coreferential noun phrases in a number of domains. The results generally agree with the findings of Nguyen and Kim (2008): biomedical text is on average 200 times less likely than news text to use gendered pronouns and twice as likely to use anaphoric definite noun phrases. At the domain level, however, there is clear variation within the biomedical corpus. In contrast to Nguyen and Kim's observations about GENIA some domains do make non-negligible use of gendered pronouns, most notably Ethics (usually to refer to other scholars) and domains such as Psychiatry and Pediatrics where studies of actual patients are common. All biomedical domains use demonstrative NPs more frequently than newswire and only one (Ethics) matches newswire for frequent use of neuter 3rd-person pronouns.

6 Conclusion

In this paper we have explored the phenomenon of linguistic variation at a finer-grained level than previous NLP research, focusing on subdomains

Pronouns (neuter, 3rd)		Pronouns (non-neuter, 3rd)		Demonstrative NPs	
Ethics	0.0658	Newswire	0.0591	Genetics	0.0275
Newswire	0.0607	Ethics	0.0037	Med. Informatics	0.0263
Therapeutics	0.0354	Pediatrics	0.0015	Biochemistry	0.0263
Med. Informatics	0.0346	Psychiatry	0.0009	Ethics	0.0260
Psychiatry	0.0342	Comm. Diseases	0.0009	Mol. Biology	0.0251
Pediatrics	0.0308	Therapeutics	0.0005	PMC Average	0.0226
PMC Average	0.0284	PMC Average	0.0005	Cell Biology	0.0210
Genetics	0.0275	Critical Care	0.0004	Comm. Diseases	0.0207
Critical Care	0.0272	Neoplasms	0.0002	Neoplasms	0.0205
Mol. Biology	0.0258	Med. Informatics	0.0002	Psychiatry	0.0201
Biochemistry	0.0251	Genetics	0.0001	Critical Care	0.0201
Neoplasms	0.0227	Mol. Biology	2.5×10^{-5}	Therapeutics	0.0192
Cell Biology	0.0217	Biochemistry	2.0×10^{-5}	Pediatrics	0.0191
Comm. Diseases	0.0213	Cell Biology	1.5×10^{-5}	Newswire	0.0118

Table 3: Frequency of coreferential types (proportion of all NPs) across domains

rather than traditional domains such as "newswire" and "biomedicine". We have identified patterns of variation across dimensions of vocabulary, syntax and discourse that are known to be of importance for NLP applications. While the magnitude of variation between subdomains is unsurprisingly less pronounced than between coarser domains, subdomain variation clearly does exist and should be taken into account when considering the generalisability of systems trained and evaluated on specific subdomains, for example molecular biology.

Future work includes directly evaluating the effect of subdomain variation on practical tasks, investigating further dimensions of variation such as nominalisation usage and learning alternative subdomain taxonomies directly from the corpus text. Ultimately, we expect that a more nuanced understanding of subdomain effects will have tangible benefits for many applications of scientific language processing.

Acknowledgements

This work was supported by EPSRC grant EP/G051070/1, the Royal Society (AK) and a Dorothy Hodgkin Postgraduate Award (LS).

References

Biber, Douglas and Bethany Gray. 2010. Challenging stereotypes about academic writing: Complexity, elaboration, explicitness. *Journal of English for Academic Purposes*, 9(1):2–20.

Biber, Douglas. 1988. *Variation Across Speech and Writing*. Cambridge University Press, Cambridge.

Blei, David M., Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Cohen, K. Bretonnel, Martha Palmer, and Lawrence Hunter. 2008. Nominalization and alternations in biomedical language. *PLoS ONE*, 3(9):e3158.

Collins, Michael John. 1996. A new statistical parser based on bigram lexical dependencies. In *Proceedings of ACL-96*, Santa Cruz, CA.

Collins, Michael. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.

Curran, James, Stephen Clark, and Johan Bos. 2007. Linguistically motivated large-scale NLP with C&C and Boxer. In *Proceedings of the ACL-07 Demo and Poster Sessions*, Prague, Czech Republic.

Daumé III, Hal and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26:101–126.

Friedman, Carol, Pauline Kraa, and Andrey Rzhetsky. 2002. Two biomedical sublanguages: a description based on the theories of Zellig Harris. *Journal of Biomedical Informatics*, 35(4):222–235.

Graff, David, Junbo Kong, Ke Chen, and Kazuaki Maeda, 2005. *English Gigaword Corpus, 2nd Edition*. Linguistic Data Consortium.

Hara, Tadayoshi, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Adapting a probabilistic disambiguation model of an HPSG parser to a new domain. In *Proceedings of IJCNLP-05*, Jeju Island, South Korea.

Jin, Yang, Ryan T. McDonald, Kevin Lerman, Mark A. Mandel, Steven Carroll, Mark Y. Liberman, Fernando C. Pereira, Raymond S. Winters, and Peter S. White. 2006. Automated recognition of malignancy mentions in biomedical literature. *BMC Bioinformatics*, 7:492.

Kim, J.-D., T. Ohta, Y. Tateisi, and J. Tsujii. 2003. GENIA corpus - a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(Suppl. 1):i180–i182.

Korhonen, Anna, Yuval Krymolowski, and Nigel Collier. 2008. The choice of features for classification of verbs in biomedical texts. In *Proceedings of COLING-08*, Manchester, UK.

Kulick, Seth, Ann Bies, Mark Liberman, Mark Mandel, Ryan McDonald, Martha Palmer, Andrew Schein, Lyle Ungar, Scott Winters, and Pete White. 2004. Integrated annotation for biomedical information extraction. In *Proceedings of the HLT-NAACL-04 Workshop on Linking Biological Literature, Ontologies and Databases*, Boston, MA.

Lappin, Shalom and Herbert J. Leass. 1994. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535–561.

Nguyen, Ngan L.T. and Jin-Dong Kim. 2008. Exploring domain differences for the design of a pronoun resolution system for biomedical text. In *Proceedings of COLING-08*, Manchester, UK.

NIH. 2009a. Journal publishing tag set. http://dtd.nlm.nih.gov/publishing/.

NIH. 2009b. National library of medicine: Journal subject terms. http://wwwcf.nlm.nih.gov/serials/journals/index.cfm.

Preiss, Judita, E.J. Briscoe, and Anna Korhonen. 2007. A system for large-scale acquisition of verbal, nominal and adjectival subcategorization frames from corpora. In *Proceedings of ACL-07*, Prague, Czech Republic.

Rayson, Paul and Roger Garside. 2000. Comparing corpora using frequency profiling. In *Proceedings of the ACL-00 Workshop on Comparing Corpora*, Hong Kong.

Rimell, Laura and Stephen Clark. 2009. Porting a lexicalized-grammar parser to the biomedical domain. *Journal of Biomedical Informatics*, 42(5):852–865.

Roland, Douglas and Daniel Jurafsky. 1998. How verb subcategorization frequencies are affected by corpus choice. In *Proceedings of COLING-ACL-98*, Montreal, Canada.

Rosario, Barbara and Marti Hearst. 2001. Classifying the semantic relations in noun compounds via a domain-specific lexical hierarchy. In *Proceedings of EMNLP-01*, Pittsburgh, PA.

Schuman, Jonathan and Sabine Bergler. 2006. Postnominal prepositional phrase attachment in proteomics. In *Proceedings of the HLT-NAACL-06 BioNLP Workshop on Linking Natural Language and Biology*, New York, NY.

Verspoor, Karin, K Bretonnel Cohen, and Lawrence Hunter. 2009. The textual characteristics of traditional and Open Access scientific journals are similar. *BMC Bioinformatics*, 10:183.

Semantic Role Labeling for News Tweets

[1,2]Xiaohua Liu, [3]Kuan Li[*], [4]Bo Han[*], [2]Ming Zhou,
[2]Long Jiang, [3]Zhongyang Xiong and [2]Changning Huang
[1]School of Computer Science and Technology
Harbin Institute of Technology
[2]Microsoft Research Asia
[3]College of Computer Science
Chongqing University
[4]School of Software
Dalian University of Technology
{xiaoliu, v-kuli, v-bohan, mingzhou, longj}
@microsoft.com
zyxiong@cqu.edu.cn
v-cnh@microsoft.com

Abstract

News tweets that report what is happening have become an important real-time information source. We raise the problem of Semantic Role Labeling (SRL) for news tweets, which is meaningful for fine grained information extraction and retrieval. We present a self-supervised learning approach to train a domain specific SRL system to resolve the problem. A large volume of training data is automatically labeled, by leveraging the existing SRL system on news domain and content similarity between news and news tweets. On a human annotated test set, our system achieves state-of-the-art performance, outperforming the SRL system trained on news.

1 Introduction

Tweets are text messages up to 140 characters. Every day, more than 50 million tweets are generated by millions of Twitter users. According to the investigation by Pear Analytics (2009), about 4% tweets are related to news[1].

We divide news related tweets into two categories: those excerpted from news articles and those not. The former kind of tweets, hereafter called news excerpt, is formally written while the latter, hereafter called news tweet, varies in style and often is not grammatically correct. To understand the proportion of news tweets, we randomly selected 1000 tweets related to news, and got 865 news tweets. Following is an example of anews tweet, containing *oh, yea,* which usually appear in spoken language, and *:-(,* an emoticon.

oh yea and Chile earthquake the earth off it's axis according to NASA and shorten the day by a wee second :-((S1)

News tweets arean important information source because they keep reporting what is happening in real time. For example, the earthquake near Los Angeles that happened on Tuesday, July 29, 2008 was first reported through news tweets only seconds later than the outbreak of the quake. Official news did not emerge about this event until four minutes later. By then, "Earthquake" was trending on Twitter Search with thousands of updates[2].

However, it is a daunting task for people to find out information they are interested in from such a huge number of news tweets, thus motivating us to conduct some kind of information

[*] This work has been done while the author was visiting Microsoft Research Asia.

[1] http://blog.twitter.com/2010/02/measuring-tweets.html

[2] http://blog.twitter.com/2008/07/twitter-as-news-wire.html

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 698–706,
Beijing, August 2010

extraction such as event mining, where SRL plays a crucial role (Surdeanu et al., 2003). Considering Sentence 1, suppose the agent *earthquake* and the patient *day* for the predicate *shorten* are identified. Then it is straightforward to output the event *Chile earthquake shorten the day*, which captures the essential information encoded in this tweet.

Following Màrquez (2009), we define SRL for news tweets as the task of identifying the arguments of a given verb as predicate in a news tweet and assigning them semantic labels describing the roles they play for the predicate. To make our method applicable to general information extraction tasks, rather than only to some special scenarios such as arresting event extraction, we adopt general semantic roles, i.e., Agent(*A0*), Patient(*A1*), Location(*AM-LOC*), Temporal(*AM-TMP*),etc., instead of situation-specific roles (Fillmore et al., 2004) such as Suspect, Authorities, and Offense in an arrest frame.

Our first attempt is to directly apply the state-of-art SRL system (Meza-Ruiz and Riedel, 2009) that trained on the CoNLL 08 shared task dataset(Surdeanu et al., 2008), hereafter called SRL-BS, to news tweets. Not surprisingly, we observe its F1 score drops sharply from 75.5% on news corpus to 43.3% on our human annotated news tweets, owing much to the informal written style of news tweets.

Therefore, we have to build a domain specific SRL system for news tweets. Given the diversified styles of news tweets, building such a system requires a larger number of annotated news tweets, which are not available, and are not affordable for human labeling. We propose a novel method to automatically annotate news tweets, which leverages the existing resources of SRL for news domain, and content similarity between news and news tweets. We argue that the same event is likely to be reported by both news and news tweets, which results in content similarity between the news and news tweet. Further, we argue that the news and news tweets reporting the same event tend to have similar predicate-argument structures. We tested our assumptions on the event *Chile earthquake* that happened on Match 2nd, 2010. We got 261 news and 722 news tweets published on the same day that described this event. Sentence 2 and 3 are two examples of the news excerpts and Sentence 1 is one example of news tweets for this event.

Chile Earthquake Shortened Earth Day (S2)

Chile Earthquake Shortened Day (S3)

Obviously Sentence 1, 2 and 3 all have predicate "*shortened"* with the same A0 and A1 arguments. Our manually checking showed that in average each news tweet in those 993 samples had 2.4 news excerpts that had the same predicate-argument structures.

Our news tweet annotation approach consists of four steps. First, we submit hot queries to Twitter and for each query we obtain a list of tweets. Second, for each list of tweets, we single out news excerpts using heuristic rules and remove them from the list, conduct SRL on news excerpts using SRL-BS, and cluster them in terms of the similarity in content and predicate-argument structures. Third, for each list of tweets, we try to merge every remaining tweet into one news excerpt cluster according to its content similarity to the cluster. Those that can be put into one news group are regarded as news tweet. Finally, semantic structures of news excerpts are passed to the news tweet in the same group through word alignment.

Our domain specific SRL system is then trained on automatically constructed training data using the Conditional Random Field (CRF: Lafferty et al., 2001) learning framework. Our system is evaluated on a human labeled dataset, and achieves state-of-the-art performance, outperforming the baseline SRL-BS.

Our contributions can be summarized as follows:

1) We propose to conduct SRL for news tweets for fine grained information extraction and retrieval;
2) We present a semi-supervised learning approach to train a domain specific SRL system for news tweets, which outperforms SRL-BS and achieves the state-of-the-art performance on a human labeled dataset.

The rest of this paper is organized as follows: In the next section, we review related work. In Section 3 we detail key components of our approach. In Section 4, we setup experiments and evaluate the effectiveness of our method. Final-

ly, Section 5 concludes and presents the future work.

2 Related Work

Our related work falls into two categories: SRL on news and domain adaption.

As for SRL on news, most researchers used the pipelined approach, i.e., dividing the task into several phases such as argument identification, argument classification, global inference, etc., and conquering them individually (Xue and Palmer, 2004; Koomen et al., 2005; Cohn and Blunsom, 2005; Punyakanok et al., 2008; Toutanova et al., 2005; Toutanova et al., 2008). Exceptions to the pipelined approach exist. Màrquez et al. (2005) sequentially labeled the words according to their positions relative to an argument (i.e., inside, outside or at the beginning of it). Carreras et al. (2004) and Surdeanu et al. (2007) jointly labeled all the predicates. Vickrey and Koller(2008) simplified the input sentence by hand-written and machine learnt rules before conducting SRL. Some other approaches simultaneously resolved all the sub-tasks by integrating syntactic parsing and SRL into a single model (Musillo and Merlo, 2006; Merlo and Musillo, 2008), or by using Markov Logic Networks (MLN, Richardson and Domingos, 2005) as the learning framework (Riedel and Meza-Ruiz, 2008; Meza-Ruiz and Riedel, 2009).

All the above approaches focus on sentences from news articles or other formal documents, and depend on human annotated corpus for training. To our knowledge, little study has been carried out on SRL for news tweets.

As for domain adaption, some researchers regarded the out-of-domain data as "prior knowledge" and estimated the model parameters by maximizing the posterior under this prior distribution, and successfully applied their approach to language modeling (Bacchiani and Roark, 2003) and parsing (Roark and Bacchiani, 2003). Daumé III and Marcu (2006) presented a novel framework by defining a "general domain" between the "truly in-domain" and "truly out-of-domain".

Unlike existing domain adaption approaches, our method is about adapting SRL system on news domain to the news tweets domain, two domains that differ in writing style but are linked through content similarity.

3 Our Method

Our method of SRL for news tweets is to train a domain specific SRL on automatically annotated training data as briefed in Section 1.

In this section we present details of the five crucial components of our method, i.e., news excerpt identification, news excerpt clustering, news tweets identification, semantic structure mapping, and the domain specific SRL system constructing.

3.1 News Excerpt Identification

We use one heuristic rule to decide whether or not a tweet is news excerpt: if a tweet has a link to a news article and its text content is included by the news article, it is news excerpt, otherwise not.

Given a tweet, to apply this rule, we first extract the content link and expand it, if any, into the full link with the unshorten service[3]. This step is necessary because content link in tweet is usually shortened to reduce the total amount of characters. Next, we check if the full link points to any of the pre-defined news sites, which, in our experiments, are 57 English news websites. If yes, we download the web page and check if it exactly contains the text content of the input tweet. Figure 1 illustrates this process.

Figure 1. An illustration of news excerpt identification.

To test the precision of this approach, while preparing for the training data for the experiments, we checked 100 tweets that were identified as news excerpt by this rule to find out they all are excerpted from news.

[3] http://unshort.me

3.2 News Excerpt Clustering

Given as input a list of news excerpts concerning the same query and published in the same time scope, this component uses the hierarchical agglomerative clustering algorithm (Manning et al., 2008) to divide news excerpts into groups in terms of the similarity in content and predicate-argument structures.

Before clustering, for every news excerpt, we remove the content link and other metadata such as author, retweet marks (starting with RT @), reply marks (starting with @ immediately after the author), hash tags (starting with #), etc., and keep only the text content; then it is further parsed into tokens, POS tags, chunks and syntactic tree using the OpenNLP toolkit[4]. After that, SRL is conducted with SRL-BS to get predicate-argument structures. Finally, every news excerpt is represented as frequency a vector of terms, including tokens, POS tagger, chunks, predicate-argument structures, etc. A news cluster is regarded as a "macro" news excerpt and is also represented as a term frequency vector, i.e., the sum of all the term vectors in the cluster. Noisy terms, such as numbers and predefined stop words are excluded from the frequency vector. To reduce data sparseness, words are stemmed by Porter stemmer (Martin F. Porter, 1980).

The cosine similarity is used to measure the relevance between two clusters, as defined in Formula 1.

$$CS(C,C') = \frac{CV \cdot CV'}{\|CV\| \times \|CV'\|} \quad (1)$$

Where *C, C'* denote two clusters, *CV, CV'* denote the term frequency vectors of *C* and *C'* respectively, and *CS(C, C')* stands for the cosine similarity between *C* and *C'*.

Initially, one news excerpt forms one cluster. Then the clustering process repeats merging the two most similar clusters into one till the similarity between any pair of clusters is below a threshold, which is experimentally set to 0.7 in our experiments.

During the training data preparation process, we randomly selected 100 clusters, each with 3.2 pieces of news in average. For every pair of news excerpts in the same cluster, we checked if they shared similar contents and semantic structures, and found out that 91.1% were the cases.

[4] http://opennlp.sourceforge.net/

3.3 News Tweets Identification

After news excerpts are identified and removed from the list, every remaining tweet is checked if it is a news tweet. Here we group news excerpts and news tweets together in two steps because 1) news excerpts count for only a small proportion of all the tweets in the list, making our two-step clustering algorithm more efficient; and 2) one-step clustering tends to output meaningless clusters that include no news tweets.

Intuitively, news tweet, more often than not, have news counterparts that report similar contents. Thus we use the following rule to identify news tweets: if the content similarity between the tweet and any news excerpt cluster is greater than a threshold, which is experimentally set to 0.7 in our experiments, the tweet is a news tweet, otherwise it is not. Furthermore, each news tweet is merged into the cluster with most similar content. Finally, we re-label any news tweet as news excerpt, which is then process by SRL-BS, if its content similarity to the cluster exceeds a threshold, which is experimentally set to 0.9 in our experiments.

Again, the cosine similarity is used to measure the content similarity between tweet and news excerpt cluster. Each tweet is repressed as a term frequency vector. Before extracting terms from tweet, tweet metadata is removed and a rule-based normalization process is conducted to restore abnormal strings, say "'", into their human friendly form, say '' ''. Next, stemming tools and OpenNLP are applied to get lemmas, POS tags, chunks, etc., and noisy terms are filtered.

We evaluated the performance of this approach when preparing for the training data. We randomly sampled 500 tweets that were identified as news tweets, to find that 93.8% were true news tweets.

3.4 Semantic Structure Mapping

Semantic structure mapping is formed as the task of word alignment from news excerpt to news tweet. A HMM alignment model is trained with GIZA++ (Franz and Hermann, 2000) on all (news excerpt, news tweet) pairs in the same cluster. After word alignment is done, semantic

information attached to a word in a news excerpt is passed to the corresponding word in the news tweet as illustrated in Figure 2.

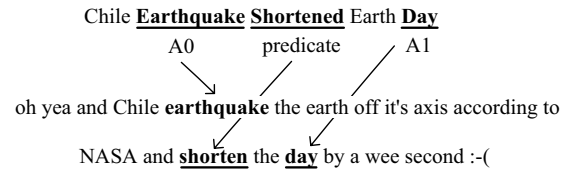


Figure 2. An example of mapping semantic structures from news excerpts to news tweets.

In Figure 2, *shorten*, *earthquake* and *day* in two sentences are aligned, respectively; and two predicate-argument structures in the first sentence, i.e., *(shortened, earthquake, A0), (shortened, day, A1)*, are passed to the second.

News tweets may receive no semantic information from related news excerpts after mapping, because of word alignment errors or no news excerpt in the cluster with similar semantic structures. Such tweets are dropped.

Mapping may also introduce cases that violate the following two structural constraints in SRL (Meza-Ruiz and Riedel, 2009): 1) one (predicate, argument) pair has only one role label in one sentence; and 2) for each predicate, each of the proper arguments *(A0~A5)* can occur at most once. Those conflicts are largely owing to the noisy outputs of SRL trained on news and to the alignment errors. While preparing for the training data for our experiments, we found 38.9% of news tweets had such conflicts.

A majority voting schema and the structural constrains are used to resolve the conflicts as described below.

1) Step 1, for every cluster, each (*predicate*, *argument*, *role*) is weighted according to its frequency in the cluster;
2) Step 2, for every cluster, detect conflicts using the structural constrains; if no conflicts exist, stop; otherwise go to Step 3;
3) Step 3, for every cluster, keep the one with higher weight in each conflicting (*predicate*, *argument*, *role*) pair; if the weights are equal, drop both;

Here is an example to show the conflicting resolution process. Consider the cluster including Sentence 1, 2 and 3, where *(shorten, earthquake, A0), (shorten, earthquake, A1), (shorten, axis, A0),* and *(shorten, day, A1)* occur 6, 4, 1 and 3 times, respectively. This cluster includes three conflicting pairs:

1) *(shorten, earthquake, A0) vs. (shorten, earthquake, A1);*
2) *(shorten, earthquake, A1) vs. (shorten, day, A1);*
3) *(shorten, earthquake, A0) vs. (shorten, axis, A0);*

The first pair is first resolved, causing *(shorten, earthquake, A0)* to be kept and *(shorten, earthquake, A1)* removed, which leads to the second pair being resolved as well; then we process the third pair resulting in *(shorten, earthquake, A0)* being kept and *(shorten, axis, A0)* dropped; finally *(shorten, earthquake, A0)* and *(shorten, day, A1)* stay in the cluster.

The conflicting resolution algorithm is sensitive to the order of conflict resolution in Step 3. Still consider the three conflicting pairs listed above. If the second pair is first processed, only *(shorten, earthquake, A0)* will be left. Our strategy is to first handle the conflict resolving which leads to most conflicts resolved.

We tested the performance of this semantic structure mapping strategy while preparing for the training data. We randomly selected 56 news tweets with conflicts and manually annotated them with SRL. After the conflict resolution method was done, we observed that 38 news tweets were resolved correctly, 9 resolved but incorrectly, and 9 remain unresolved, suggesting the high precision of this method, which fits our task. We leave it to our future work to study more advanced approach for semantic structure mapping.

3.5 SRL System for News Tweets

Following Màrquez et al. (2005), we regard SRL for tweets as a sequential labeling task, because of its joint inference ability and its openness to support other languages.

We adopt conventional features for each token defined in Màrquez et al.(2005), such as the lemma/POS tag of the current/previous/next token, the lemma of predicate and its combination with the lemma/POS tag of the current token, the voice of the predicate (active/passive), the distance between the current token and the predicate, the relative position of the current token to

the predicate, and so on. We do not use features related to syntactic parsing trees, to allow our system not to rely on any syntactic parser, whose performance depends on style and language of text, which limits the generality of our system.

Before extracting features, we perform a preprocessing step to remove tweet metadata and normalize tweet text content, as described in Section 3.3. The OpenNLP toolkit is used for feature extraction, and the CRF++ toolkit[5] is used to train the model.

4 Experiments

In this section, we evaluate our SRL system on a gold-standard dataset consisting of 1,110 human annotated news tweets and show that our system achieves the state-of-the-art performance compared with SRL-BS that is trained on news. Furthermore, we study the contribution of automatically generated training data.

4.1 Evaluation Metric

We adopt the widely used precision (Pre.), recall (Rec.) and F-score (F., the harmonic mean of precision and recall) as evaluation metrics.

4.2 Baseline System

We use SRL-BS as our baseline because of its state-of-art performance on news domain, and its readiness to use as well.

4.3 Data Preparation

We restrict to English news tweets to test our method. Our method can label news tweets of other languages, given that the related tools such as the SRL system on news domain, the word alignment tool, OpenNLP, etc., can support other languages.

We build two corpora for our experiments: one is the training dataset of 10,000 news tweets with semantic roles automatically labeled; the other is the gold-standard dataset of 1,110 news tweets with semantic roles manually labeled.

Training Dataset

We randomly sample 80 queries from 300 English queries extracted from the top stories of Bing news, Google news and Twitter trending topics from March 1, 2010 to March 4, 2010.

Submitting the 80 queries to Twitter search, we retrieve and download 512,000 tweets, from which we got 4,785 news excerpts and 11,427 news tweets, which were automatically annotated using the method described in Section 3.

Furthermore, 10,000 tweets are randomly selected from the automatically annotated news tweets, forming the training dataset, while the other 1,427 news tweets are used to construct the gold-standard dataset.

Gold-standard Dataset

We ask two people to annotate the 1,427 news tweets, following the Annotation guidelines for PropBank[6] with one exception: for phrasal arguments, only the head word is labeled as the argument, because our system and SRL-BS conduct word level SRL.

317 news tweets are dropped because of inconsistent annotation, and the remaining 1,110 news tweets form the gold-standard dataset.

Quality of Training dataset

Since the news tweets in the gold-standard dataset are randomly sampled from the automatically labeled corpus and are labeled by both human and machine, we use them to estimate the quality of training data, i.e., to which degree the automatically generated results are similar to humans'.

We find that our method achieves 75.6% F1 score, much higher than the baseline, suggesting the relatively high quality of the training data.

4.4 Result and Analysis

Table 1 reports the experimental results of our system (SRL-TS) and the baseline on the gold-standard dataset.

	Precision	Recall	F-Score
SRL-BS	36.0 %	54.5%	43.3%
SRL-TS	78.0%	57.1%	66.0%

Table 1. Performances of our system and the baseline on the gold-standard dataset.

As shown in Table 1, our system performs much better than the baseline on the gold-standard dataset in terms of all metrics. We observe two types of errors that are often made by

[5] http://crfpp.sourceforge.net/

[6] http://verbs.colorado.edu/~mpalmer/projects/ace/PB guidelines.pdf

SRL-BS but not so often by our system, which largely explains the difference in performance.

The first type of errors, which accounts for 25.3% of the total errors made by SRL-BS, is caused by the informal written style, such as ellipsis, of news tweets. For instance, for the example Sentence 1 listed in Section 1, the SRL-BS incorrectly identify earth as the A0 argument of the predicate shorten. The other type of errors, which accounts for 10.2% of the total errors made by SRL-BS, is related to the discretionary combination of news snippets. For example, consider the following news tweet:

The Chile earthquake shifted the earth's axis, "shortened the length of an Earth day by 1.26 miliseconds". (S4)

We analyze the errors made by our system and find that 12.5% errors are attributed to the complex syntactic structures, suggesting that combining our system with systems on news domain is a promising direction. For example, our system cannot identify the A0 argument of the predicate *shortened*, because of its blindness of attributive clause; in contrast, SRL-BS works on this case.

wow..the earthquake that caused the 2004 Indian Ocean tsunami shortened the day by almost 3 microseconds..what does that even mean?! HOW? (S5)

We also find that 32.3% of the errors made by our system are more or less related to the training data, which has noise and cannot fully represent the knowledge of SRL on news tweets. For instance, our system fails to label the following sentence, partially because the predicate *strike* does not occur in the training set.

8.8-Magnitude-Earthquake-Strikes-Chile (S6)

We further study how the size of automatically labeled training data affects our system's performance, as illustrated in Figure 3. We conduct two sets of experiments: in the first set, the training data is automatically labeled and the testing data is the gold-standard dataset; in the second set, half of the news tweets from the gold-standard dataset are added to the training data, the remaining half forms the testing dataset. Curve 1 and 2 represent the experimental results of set 1 and 2, respectively.

From Curve 1, we see that our system's performance increases sharply when the training data size varies from 5,000 to 6,000; then increases relatively slowly with more training data; and finally reaches the highest when all training data is used. Curve 2 reveals a similar trend.

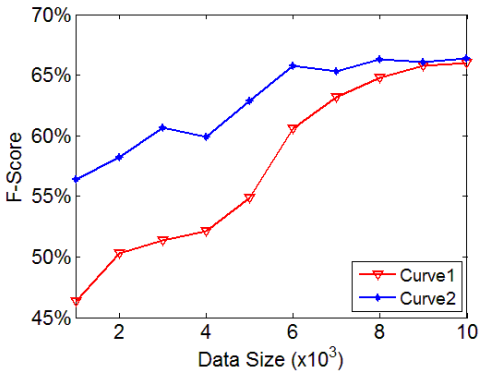


Figure 3. Performance on training data of varying size.

This phenomenon is largely due to the competing between two forces: the noise in the training data, and the knowledge of SRL encoded in the training data.

Interestingly, from Figure 3, we observe that the contribution of human labeled data is no longer significant after 6,000 automatically labeled training data is used, reaffirming the effectiveness of the training data.

5 Conclusions and Future Work

We propose to conduct SRL on news tweets for fine grained information extraction and retrieval. We present a self-supervised learning approach to train a domain specific SRL system for news tweets. Leveraging the SRL system on news domain and content similarity between news and news tweets, our approach automatically labels a large volume of training data by mapping SRL-BS generated results of news excerpts to news tweets. Experimental results show that our system outperforms the baseline and achieves the state-of-the-art performance.

In the future, we plan to enlarge training data size and test our system on a larger dataset; we also plan to further boost the performance of our system by incorporating tweets specific features such as hash tags, reply/re-tweet marks into our

CRF model, and by combining our system with SRL systems trained on news.

References

Bacchiani, Michiel and Brian Roark. 2003. Unsupervised language model adaptation. *Proceedings of the 2003 International Conference on Acoustics, Speech and Signal Processing,* volume 1, pages: 224-227

Carreras, Xavier, Lluís Màrquez, and Grzegorz Chrupała. 2004. Hierarchical recognition of propositional arguments with Perceptrons. *Proceedings of the Eighth Conference on Computational Natural Language Learning*, pages: 106-109.

Cohn, Trevor and Philip Blunsom. 2005. Semantic role labeling with tree conditional random fields. *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages: 169-172.

Daumé, Hal III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26(1), 101-126.

Fillmore, Charles J., Josef Ruppenhofer, Collin F. Baker. 2004. FrameNet and Representing the Link between Semantic and Syntactic Relations. *Computational Linguistics and Beyond, Institute of Linguistics, Academia Sinica.*

Kelly, Ryan, ed. 2009. *Twitter Study Reveals Interesting Results About Usage*. San Antonio, Texas: Pear Analytics.

Koomen, Peter, Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2005. Generalized inference with multiple semantic role labeling systems. *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages: 181-184.

Lafferty, John D., Andrew McCallum, Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. *Proceedings of the Eighteenth International Conference on Machine Learning*, pages: 282-289.

Manning, Christopher D., Prabhakar Raghavan and Hinrich Schtze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK.

Màrquez, Lluís, Jesus Gimènez Pere Comas and Neus Català. 2005. Semantic Role Labeling as Sequential Tagging. *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages: 193-196.

Màrquez, Lluís. 2009. *Semantic Role Labeling Past, Present and Future*, Tutorial of ACL-IJCNLP 2009.

Merlo, Paola and Gabriele Musillo. 2008. Semantic parsing for high-precision semantic role labelling. *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages: 1-8.

Meza-Ruiz, Ivan and Sebastian Riedel. 2009. Jointly Identifying Predicates, Arguments and Senses using Markov Logic. *Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL*, pages: 155-163.

Musillo, Gabriele and Paola Merlo. 2006. Accurate Parsing of the proposition bank. *Proceedings of the Human Language Technology Conference of the NAACL*, pages: 101-104.

Och, Franz Josef, Hermann Ney. Improved Statistical Alignment Models. *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages: 440-447.

Porter, Martin F. 1980. An algorithm for suffix stripping. *Program,* 14(3), 130-137.

Punyakanok, Vasin, Dan Roth and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Journal of Computational Linguistics*, 34(2), 257-287.

Richardson, Matthew and Pedro Domingos. 2005. Markov logic networks. *Technical Report, University of Washington,* 2005.

Riedel, Sebastian and Ivan Meza-Ruiz. 2008. Collective semantic role labelling with Markov Logic. *Proceedings of the Twelfth Conference on Computational Natural Language Learning,* pages: 193-197.

Roark, Brian and Michiel Bacchiani. 2003. Supervised and unsupervised PCFG adaptation to novel domains. *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology,* volume 1, pages: 126-133.

Surdeanu, Mihai, Sanda Harabagiu, JohnWilliams and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics,* volume 1, pages: 8-15.

Surdeanu, Mihai, Lluís Màrquez, Xavier Carreras and Pere R. Comas. 2007. Combination strategies for semantic role labeling. *Journal of Artificial Intelligence Research,* 29(1), 105-151.

Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The conll 2008 shared task on joint parsing of syntactic and semantic dependencies. *Proceedings of the Twelfth Conference on Computational Natural Language Learning,* pages: 159-177.

Toutanova, Kristina, Aria Haghighi and Christopher D. Manning. 2005. Joint learning improves semantic role labeling. *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics,* pages: 589-596.

Toutanova, Kristina, Aria Haghighi and Christopher D. Manning. 2008. A global joint model for semantic role labeling. *Journal of Computational Linguistics*, 34(2), 161-191.

Vickrey, David and Daphne Koller. 2008. Applying sentence simplification to the conll-2008 shared task. *Proceedings of the Twelfth Conference on Computational Natural Language Learning,* pages: 268-272

Xue, Nianwen and Martha Palmer. 2004. Calibrating features for semantic role labeling. *Proceedings of the Conference on Empirical Methods in Natural Language Processing,* pages: 88-94.

Joint Parsing and Translation

Yang Liu and **Qun Liu**
Key Laboratory of Intelligent Information Processing
Institute of Computing Technology
Chinese Academy of Sciences
{yliu,liuqun} @ict.ac.cn

Abstract

Tree-based translation models, which exploit the linguistic syntax of source language, usually separate decoding into two steps: parsing and translation. Although this separation makes tree-based decoding simple and efficient, its translation performance is usually limited by the number of parse trees offered by parser. Alternatively, we propose to parse and translate jointly by casting tree-based translation as parsing. Given a source-language sentence, our joint decoder produces a parse tree on the source side and a translation on the target side simultaneously. By combining translation and parsing models in a discriminative framework, our approach significantly outperforms a forest-based tree-to-string system by 1.1 absolute BLEU points on the NIST 2005 Chinese-English test set. As a parser, our joint decoder achieves an F_1 score of 80.6% on the Penn Chinese Treebank.

1 Introduction

Recent several years have witnessed the rapid development of syntax-based translation models (Chiang, 2007; Galley et al., 2006; Shen et al., 2008; Quirk et al., 2005; Liu et al., 2006; Huang et al., 2006; Eisner, 2003; Zhang et al., 2008; Chiang, 2010), which incorporate formal or linguistic syntax into translation process. Depending on whether modeling the linguistic syntax of source language or not, we divide them into two categories: *string-based* and *tree-based* models. [1]

[1]Mi et al. (2008) also distinguish between string-based and tree-based models but depending on the type of input.

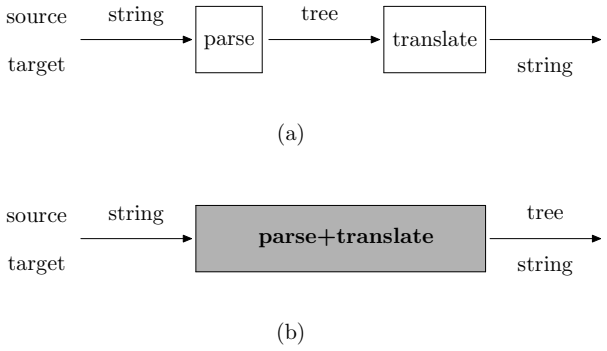


Figure 1: Tree-based decoding: (a) separate parsing and translation versus (b) joint parsing and translation.

String-based models include *string-to-string* (Chiang, 2007) and *string-to-tree* (Galley et al., 2006; Shen et al., 2008). Regardless of the syntactic information on the source side, they treat decoding as a parsing problem: the decoder parses a source-language sentence using the source projection of a synchronous grammar while building the target sub-translations in parallel.

Tree-based models include *tree-to-string* (Liu et al., 2006; Huang et al., 2006) and *tree-to-tree* (Quirk et al., 2005; Eisner, 2003; Zhang et al., 2008; Chiang, 2010). These models explicitly use source parse trees and divide decoding into two separate steps: parsing and translation. A parser first parses a source-language sentence into a parse tree, and then a decoder converts the tree to a translation on the target side (see Figure 1(a)).

Figure 2 gives a training example for tree-to-string translation, which consists of a Chinese tree, an English sentence, and the word alignment between them. Romanized Chinese words are given to facilitate identification. Table 1 shows

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 707–715,
Beijing, August 2010

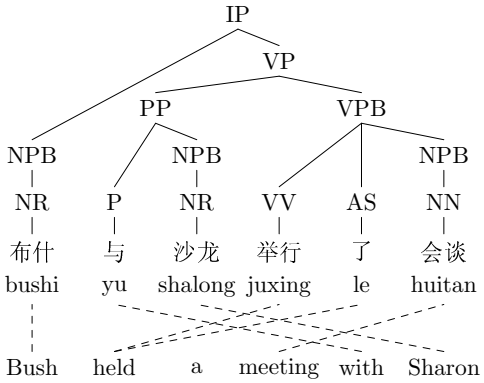


Figure 2: A training example that consists of a Chinese parse, an English sentence, and the word alignment between them.

a set of tree-to-string rules obtained from Figure 2. The source side of a rule is a tree fragment and the target side is a string. We use x to denote non-terminals and the associated subscripts indicate the correspondence between non-terminals on both sides.

Conventionally, decoding for tree-to-string translation is cast as a *tree parsing* problem (Eisner, 2003). The tree parsing algorithm visits each node in the input source tree in a top-down order and tries to match each translation rule against the local sub-tree rooted at the node. For example, the first rule in Table 1 matches a sub-tree rooted at $\mathrm{IP}_{0,6}$ in Figure 2. The descendent nodes of $\mathrm{IP}_{0,6}$ (i.e., $\mathrm{NPB}_{0,1}$, $\mathrm{PP}_{1,3}$, and $\mathrm{VPB}_{3,6}$) can be further matched by other rules in Table 1. The matching procedure runs recursively until the entire tree is covered. Finally, the output on the target side can be taken as a translation.

Compared with its string-based counterparts, tree-based decoding is simpler and faster: there is no need for *synchronous binarization* (Huang et al., 2009b; Zhang et al., 2006) and tree parsing generally runs in linear time (Huang et al., 2006).

While separating parsing and translation makes tree-based decoding simple and efficient, its search space is limited by the number of parse trees offered by parser. Studies reveal that tree-based systems are prone to produce degenerate translations due to the propagation of parsing mistakes (Quirk and Corston-Oliver, 2006). This problem can be alleviated by offering more alternatives to the pipeline. An elegant solution is to replace 1-best trees with packed forests that encode exponentially many trees (Mi et al., 2008; Liu et al., 2009). Mi et al. (2008) present an efficient algorithm to match tree-to-string rules against packed forests that encode millions of trees. They prove that offering more alternatives to tree parsing improves translation performance substantially.

(1)	IP(x_1:NPB VP(x_2:PP x_3:VPB))→x_1 x_3 x_2
(2)	NPB(NR(*bushi*))→*Bush*
(3)	PP(P(*yu*) x_1:NPB)→*with* x_1
(4)	NPB(NR(*shalong*))→*Sharon*
(5)	VPB(VV(*juxing*) AS(*le*) x_1:NPB)→*held a* x_1
(6)	NPB(NN(*huitan*))→*meeting*

Table 1: Tree-to-string rules extracted from Figure 2.

In this paper, we take a further step towards the direction of offering multiple parses to translation by proposing *joint parsing and translation*. As shown in Figure 1(b), our approach parses and translates jointly as it finds a parse tree and a translation of a source-language sentence simultaneously. We integrate the tree-to-string model (Liu et al., 2006; Huang et al., 2006), n-gram language model, probabilistic context-free grammar (PCFG), and Collins' Model 1 (Collins, 2003) in a discriminative framework (Och, 2003). Allowing parsing and translation to interact with each other, our approach obtains an absolute improvement of 1.1 BLEU points over a forest-based tree-to-string translation system (Mi et al., 2008) on the 2005 NIST Chinese-English test set. As a parser, our joint decoder achieves an F_1 score of 80.6% on the Penn Chinese Treebank.

2 Joint Parsing and Translation

2.1 Decoding as Parsing

We propose to integrate parsing and translation into a single step. To achieve joint parsing and translation, we cast tree-to-string decoding as a monolingual parsing problem (Melamed, 2004; Chiang, 2007; Galley et al., 2006): the decoder takes a source-language string as input and parses it using the source-projection of SCFG while building the corresponding sub-translations simultaneously.

For example, given the Chinese sentence *bushi yu sha long juxing le huitan* in Figure 2, the derivation in Table 1 explains how a Chinese tree, an English string, and the word alignment between them are generated synchronously. Unlike the string-based systems as described in (Chiang, 2007; Galley et al., 2006; Shen et al., 2008), we exploit the linguistic syntax on the source side explicitly. Therefore, the source parse trees produced by our decoder are meaningful from a linguistic point of view.

As tree-to-string rules usually have multiple non-terminals that make decoding complexity generally exponential, synchronous binarization (Huang et al., 2009b; Zhang et al., 2006) is a key technique for applying the CKY algorithm to parsing with tree-to-string rules. [2] Huang et al. (2009b) factor each tree-to-string rule into two SCFG rules: one from the root nonterminal to the subtree, and the other from the subtree to the leaves. In this way, one can uniquely reconstruct the original tree using a two-step SCFG derivation.

For example, consider the first rule in Table 1:

$$\text{IP}(x_1{:}\text{NPB VP}(x_2{:}\text{PP } x_3{:}\text{VB}))\rightarrow x_1\ x_3\ x_2$$

We use a specific non-terminal, say, T, to uniquely identify the left-hand side subtree and produce two SCFG rules: [3]

$$\text{IP} \rightarrow \langle \text{T}_{\boxed{1}}, \text{T}_{\boxed{1}} \rangle \quad (1)$$

$$\text{T} \rightarrow \langle \text{NPB}_{\boxed{1}}\text{PP}_{\boxed{2}}\text{VPB}_{\boxed{3}}, \text{NPB}_{\boxed{1}}\text{VPB}_{\boxed{3}}\text{PP}_{\boxed{2}} \rangle \quad (2)$$

where the boxed numbers indicate the correspondence between nonterminals.

Then, the rule (2) can be further binarized into two rules that have at most two non-terminals:

$$\text{T} \rightarrow \langle \text{NPB}_{\boxed{1}}\ \text{PP-VPB}_{\boxed{2}}, \text{NPB}_{\boxed{1}}\ \text{PP-VPB}_{\boxed{2}} \rangle \quad (3)$$

$$\text{PP-VPB} \rightarrow \langle \text{PP}_{\boxed{1}}\text{VPB}_{\boxed{2}}, \text{VPB}_{\boxed{2}}\text{PP}_{\boxed{1}} \rangle \quad (4)$$

where PP-VPB is an intermediate *virtual non-terminal*.

[2] But CKY is not the only choice. The Earley algorithm can also be used to parse with tree-to-string rules (Zhao and Al-Onaizan, 2008). As the Earley algorithm binarizes multi-nonterminal rules implicitly, there is no need for synchronous binarization.

[3] It might look strange that the node VP disappears. This node is actually stored in the monolithic node T. Please refer to page 573 of (Huang et al., 2009b) for more details about how to convert tree-to-string rules to SCFG rules.

We call rules the tree roots of which are virtual non-terminals *virtual rules* and others *natural rules*. For example, the rule (1) is a natural rule and the rules (3) and (4) are virtual rules. We follow Huang et al. (2009b) to keep the probabilities of a natural rule unchanged and set those of a virtual rule to 1. [4]

After binarizing tree-to-string rules into SCFG rules that have at most two non-terminals, we can use the CKY algorithm to parse a source sentence and produce its translation simultaneously as described in (Chiang, 2007; Galley et al., 2006).

2.2 Adding Parsing Models

As our decoder produces "genuine" parse trees during decoding, we can integrate parsing models as features together with translation features such as the tree-to-string model, n-gram language model, and word penalty into a discriminative framework (Och, 2003). We expect that parsing and translation could interact with each other: parsing offers linguistically motivated reordering to translation and translation helps parsing resolve ambiguity.

2.2.1 PCFG

We use the probabilistic context-free grammar (PCFG) as the first parsing feature in our decoder. Given a PCFG, the probability for a tree is the product of probabilities for the rules that it contains. That is, if a tree π is a context-free derivation that involves K rules of the form $\alpha_k \rightarrow \beta_k$, its probability is given by

$$\mathcal{P}(\pi) = \prod_{k=1...K} \mathcal{P}_{pcfg}(\alpha_k \rightarrow \beta_k) \quad (5)$$

For example, the probability for the tree in Figure 2 is

$$\begin{aligned} \mathcal{P}(\pi) \quad = \quad & \mathcal{P}_{pcfg}(\text{IP} \rightarrow \text{NPB VP}) \times \\ & \mathcal{P}_{pcfg}(\text{NPB} \rightarrow \text{NR}) \times \\ & \mathcal{P}_{pcfg}(\text{NR} \rightarrow \textit{bushi}) \times \\ & \ldots \end{aligned} \quad (6)$$

[4] This makes the scores of hypotheses in the same chart cell hardly comparable because some hypotheses are covered by a natural non-terminal and others covered by a virtual non-terminal. To alleviate this problem, we follow Huang et al. (2009b) to separate natural and virtual hypotheses in different beams.

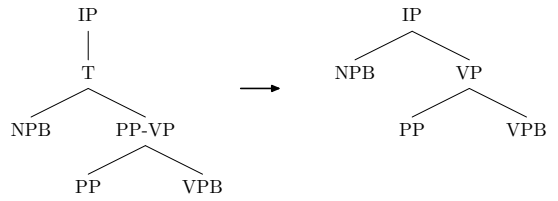


Figure 3: Reconstructing original tree from virtual rules. We first construct the tree on the left by substituting the trees of the rules (1), (3), and (4) and then restore the original tree on the right via the monolithic node T.

There are 13 PCFG rules involved. We omit the remaining 10 rules.

We formalize the decoding process as a deductive system to illustrate how to include a PCFG. Given a natural rule

$$\mathrm{VP} \rightarrow \langle \mathrm{PP}_{\boxed{1}}\ \mathrm{VPB}_{\boxed{2}}, \mathrm{VPB}_{\boxed{2}}\ \mathrm{PP}_{\boxed{1}} \rangle \quad (7)$$

the following deductive step grows an item in the chart by the rule

$$\frac{(\mathrm{PP}_{1,3}) : (w_1, e_1) \quad (\mathrm{VPB}_{3,6}) : (w_2, e_2)}{(\mathrm{VP}_{1,6}) : (w, e_2 e_1)} \quad (8)$$

where $\mathrm{PP}_{1,3}$ denotes the recognition of the non-terminal PP spanning from the substring from position 1 through 3 (i.e., *yu shalong* in Figure 2), w_1 and e_1 are the score and translation of the first antecedent item, respectively, and the resulting item score is calculated as: [5]

$$w = w_1 + w_2 + \log \mathcal{P}_{pcfg}(\mathrm{VP} \rightarrow \mathrm{PP\ VPB}) \quad (9)$$

As the PCFG probabilities of natural rules are fixed during decoding, they can be pre-computed and stored in the rule table. Therefore, including PCFG for natural rules hardly increases decoding complexity.

However, calculating the PCFG probabilities for virtual rules is quite different due to the presence of virtual non-terminals. For instance, using the rule (4) in Section 2.1 to generate an item leads to the following deductive step:

$$\frac{(\mathrm{PP}_{1,3}) : (w_1, e_1) \quad (\mathrm{VPB}_{3,6}) : (w_2, e_2)}{(\text{PP-VPB}_{1,6}) : (w, e_2 e_1)} \quad (10)$$

[5]The logarithmic form of probability is used to avoid manipulating very small numbers for practical reasons. w_1 and w_2 take the PCFG probabilities of the two antecedent items into consideration.

As PP-VPB is a virtual non-terminal, the subtree it dominates is a virtual tree, for which we cannot figure out its PCFG probability. Therefore, we have to postpone the calculation of PCFG probabilities until reaching a natural non-terminal such as IP. In other words, only when using the rule (1) to produce an item, the decoding algorithm can update PCFG probabilities because the original tree can be restored from the special node T now. Figure 3 shows how to reconstruct the original tree from virtual rules. We first construct the tree on the left by substituting the trees of the rules (1), (3), and (4) and then restore the original tree on the right via T. Now, we can calculate the PCFG probability of the original tree. [6] In practice, we pre-compute this PCFG probability and store it in the rule (1) to reduce computational overhead.

2.2.2 Lexicalized PCFG

Although widely used in natural language processing, PCFGs are often criticized for the lack of lexicalization, which is very important to capture the lexical dependencies between words. Therefore, we use Collins' Model 1 (Collins, 2003), a simple and effective lexicalized parsing model, as the second parsing feature in our decoder.

Following Collins (2003), we first lexicalize a tree by associating a *headword* h with each non-terminal. Figure 4 gives the lexicalized tree corresponding to Figure 2. The left-hand side of a rule in a lexicalized PCFG is $P(h)$ and the right-hand side has the form:

$$L_n(l_n) \ldots L_1(l_1) H(h) R_1(\tau_1) \ldots R_m(\tau_m) \quad (11)$$

where H is the head-child that inherits the headword h from its parent P, $L_1 \ldots L_n$ and $R_1 \ldots R_m$ are left and right modifiers of H, and $l_1 \ldots l_n$ and $\tau_1 \ldots \tau_m$ are the corresponding headwords. Either n or m may be zero, and $n = m = 0$ for unary rules. Collins (2003) extends the left and right sequences to include a terminating STOP symbol. Thus, $L_{n+1} = R_{m+1} = \mathrm{STOP}$.

[6]Postponing the calculation of PCFG probabilities also leads to the "hard-to-compare" problem mentioned in footnote 4 due to the presence of virtual non-terminals. We still maintain multiple beams for natural and virtual hypotheses (i.e., items) to alleviate this prblem.

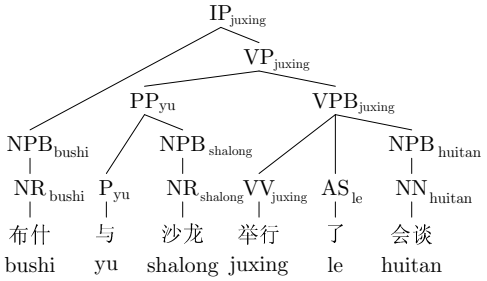


Figure 4: The lexicalized tree corresponding to Figure 2.

Collins (2003) breaks down the generation of the right-hand side of a rule into a sequence of smaller steps. The probability of a rule is decomposed as:

$$\begin{aligned}&\mathcal{P}_h(H|P(h)) \times \\ &\prod_{i=1...n+1} \mathcal{P}_l(L_i(l_i)|P(h),H,t,\Delta) \times \\ &\prod_{j=1...m+1} \mathcal{P}_r(R_j(\tau_j)|P(h),H,t,\Delta) \quad (12)\end{aligned}$$

where t is the POS tag of of the headword h and Δ is the distance between words that captures head-modifier relationship.

For example, the probability of the lexicalized rule $\mathrm{IP}(\textit{juxing}) \rightarrow \mathrm{NPB}(\textit{bushi})\ \mathrm{VP}(\textit{juxing})$ can be computed as [7]

$$\begin{aligned}&\mathcal{P}_h(\mathrm{VP}|\mathrm{IP},\ \textit{juxing}) \times \\ &\mathcal{P}_l(\mathrm{NPB}(\textit{bushi})|\mathrm{IP},\mathrm{VP},\ \textit{juxing}) \times \\ &\mathcal{P}_l(\mathrm{STOP}|\mathrm{IP},\mathrm{VP},\ \textit{juxing}) \times \\ &\mathcal{P}_r(\mathrm{STOP}|\mathrm{IP},\mathrm{VP},\ \textit{juxing}) \quad (13)\end{aligned}$$

We still use the deductive system to explain how to integrate the lexicalized PCFG into the decoding process. Now, Eq. (8) can be rewritten as:

$$\frac{(\mathrm{PP}^{\mathrm{yu}}_{1,3}) : (w_1, e_1) \quad (\mathrm{VPB}^{\mathrm{juxing}}_{3,6}) : (w_2, e_2)}{(\mathrm{VP}^{\mathrm{juxing}}_{1,6}) : (w, e_2 e_1)} \quad (14)$$

where *yu* and *juxing* are the headwords attached to $\mathrm{PP}_{1,3}$, $\mathrm{VPB}_{3,6}$, and $\mathrm{VP}_{1,6}$. The resulting item score is given by

$$\begin{aligned}w \;=\; & w_1 + w_2 + \log\mathcal{P}_h(\mathrm{VPB}|\mathrm{VP},\ \textit{juxing}) + \\ & \log\mathcal{P}_l(\mathrm{PP}(\textit{yu})|\mathrm{VP},\mathrm{VPB},\ \textit{juxing}) + \\ & \log\mathcal{P}_l(\mathrm{STOP}|\mathrm{VP},\mathrm{VPB},\ \textit{juxing}) + \\ & \log\mathcal{P}_r(\mathrm{STOP}|\mathrm{VP},\mathrm{VPB},\ \textit{juxing}) \quad (15)\end{aligned}$$

Unfortunately, the lexicalized PCFG probabilities of most natural rules cannot be pre-computed because the headword of a non-terminal must be determined on the fly during decoding. Consider the third rule in Table 1

$$\mathrm{PP}(\mathrm{P}(\textit{yu})\ x_1{:}\mathrm{NPB}) \rightarrow \textit{with}\ x_1$$

It is impossible to know what the headword of NPB is in advance, which depends on the actual sentence being translated. However, we could safely say that the headword attached to PP is always *yu* because PP should have the same headword with its child P.

Similar to the PCFG scenario, calculating lexicalized PCFG for virtual rules is different from natural rules. Consider the rule (4) in Section 2.1, the corresponding deductive step is

$$\frac{(\mathrm{PP}^{\mathrm{yu}}_{1,3}) : (w_1, e_1) \quad (\mathrm{VPB}^{\mathrm{juxing}}_{3,6}) : (w_2, e_2)}{(\text{PP-VPB}^{-}_{1,6}) : (w, e_2 e_1)} \quad (16)$$

where "$-$" denotes that the headword of $\text{PP-VPB}_{1,6}$ is undefined.

We still need to postpone the calculation of lexicalized PCFG probabilities until reaching a natural non-terminal such as IP. In other words, only when using the rule (1) to produce an item, the decoding algorithm can update the lexicalized PCFG probabilities. After restoring the original tree from T, we need to visit backwards to frontier nodes of the tree to find headwords and calculate lexicalized PCFG probabilities. More specifically, updating lexicalized PCFG probabilities for the rule the rule (1) involves the following steps:

1. Reconstruct the original tree from the rules (1), (3), and (4) as shown in Figure 3;
2. Attach headwords to all nodes;
3. Calculate the lexicalized PCFG probabilities according to Eq. (12).

[7] For simplicity, we omit POS tag and distance in the presentation. In practice, we implemented the Collins' Model 1 exactly as described in (Collins, 2003).

Back-off level	$\mathcal{P}_h(H\|\dots)$	$\mathcal{P}_l(L_i(l_i)\|\dots)$ $\mathcal{P}_r(R_j(\tau_j)\|\dots)$
1	P, h, t	P, H, h, t, Δ
2	P, t	P, H, t, Δ
3	P	P, H, Δ

Table 2: The conditioning variables for each level of back-off.

As suggested by Collins (2003), we use back-off smoothing for sub-model probabilities during decoding. Table 2 shows the various levels of back-off for each type of parameter in the lexicalized parsing model we use. For example, $\mathcal{P}_h$ estimation p interpolates maximum-likelihood estimates $p_1 = \mathcal{P}_h(H|P,h,t)$, $p_2 = \mathcal{P}_h(H|P,t)$, and $p_3 = \mathcal{P}_h(H|P)$ as follows:

$$p_1 = \lambda_1 p_1 + (1-\lambda_1)(\lambda_2 p_2 + (1-\lambda_2)p_3) \quad (17)$$

where λ_1, λ_2, and λ_3 are smoothing parameters.

3 Experiments

In this section, we try to answer two questions:

1. Does tree-based translation by parsing outperform the conventional tree parsing algorithm? (Section 3.1)

2. How about the parsing performance of the joint decoder? (Section 3.2)

3.1 Translation Evaluation

We used a bilingual corpus consisting of 251K sentences with 7.3M Chinese words and 9.2M English words to extract tree-to-string rules. The Chinese sentences in the bilingual corpus were parsed by an in-house parser (Xiong et al., 2005), which obtains an F_1 score of 84.4% on the Penn Chinese Treebank. After running GIZA++ (Och and Ney, 2003) to obtain word alignments, we used the GHKM algorithm (Galley et al., 2004) and extracted 11.4M tree-to-string rules from the source-side parsed, word-aligned bilingual corpus. Note that the bilingual corpus does not contain the bilingual version of Penn Chinese Treebank. In other words, all tree-to-string rules were learned from noisy parse trees and alignments. We used the SRILM toolkit (Stolcke, 2002) to train a 4-gram language model on the Xinhua portion of the GIGAWORD corpus, which contains 238M English words. We trained PCFG and Collins' Model 1 on the Penn Chinese Treebank.

We used the 2002 NIST MT Chinese-English test set as the development set and the 2005 NIST test set as the test set. Following Huang (2008), we modified our in-house parser to produce and prune packed forests on the development and test sets. There are about 105M parse trees encoded in a forest of a sentence on average. We also extracted 1-best trees from the forests.

As the development and test sets have many long sentences ($\geq$ 100 words) that make our decoder prohibitively slow, we divided long sentences into short sub-sentences simply based on punctuation marks such as comma and period. The source trees and target translations of sub-sentences were concatenated to form the tree and translation of the original sentence.

We compared our parsing-based decoder with the tree-to-string translation systems based on the tree parsing algorithm, which match rules against either 1-best trees (Liu et al., 2006; Huang et al., 2006) or packed forests (Mi et al., 2008). All the three systems used the same rule set containing 11.4M tree-to-string rules. Given the 1-best trees of the test set, there are 1.2M tree-to-string rules that match fragments of the 1-best trees. For the forest-based system (Mi et al., 2008), the number of filtered rules increases to 1.9M after replacing 1-best trees with packed forests, which contain 105M trees on average. As our decoder takes a string as input, 7.7M tree-to-string rules can be used to parse and translate the test set. We binarized 99.6% of tree-to-string rules into 16.2M SCFG rules and discarded non-binarizable rules. As a result, the search space of our decoder is much larger than those of the tree parsing counterparts.

Table 3 shows the results. All the three systems used the conventional translation features such as relative frequencies, lexical weights, rule count, n-gram language model, and word count. Without any parsing models, the tree-based system achieves a BLEU score of 29.8. The forest-based system outperforms the tree-based system by $+1.8$ BLEU points. Note that each hyperedge

Algorithm	Input	Parsing model	# of rules	BLEU (%)	Time (s)
tree parsing	tree	-	1.2M	29.8	0.56
	forest	PCFG	1.9M	31.6	9.49
parsing	string	-	7.7M	32.0	51.41
		PCFG		32.4	55.52
		Lex		32.6	89.35
		PCFG + Lex		**32.7**	91.72

Table 3: Comparison of tree parsing and parsing for tree-to-string translation in terms of *case-insensitive* BLEU score and average decoding time (second per sentence). The column "parsing model" indicates which parsing models were used in decoding. We use "-" to denote using only translation features. "Lex" represents the Collins' Model 1. We excluded the extra parsing time for producing 1-best trees and packed forests.

Forest size	Exact match (%)	Precision (%)
1	0.55	41.5
390	0.74	47.7
5.8M	0.92	54.1
66M	1.48	62.0
105M	2.22	65.9

Table 4: Comparison of 1-best trees produced by our decoder and the parse forests produced by the monolingual Chinese parser. Forest size represents the average number of trees stored in a forest.

in a parse forest is assigned a PCFG probability. Therefore, the forest-based system actually includes PCFG as a feature (Mi et al., 2008). Without incorporating any parsing models as features, our joint decoder achieves a BLEU score of 32.0. Adding PCFG and Collins' Model 1 (i.e., "Lex" in Table 2) increases translation performance. When both PCFG and Collins' Model 1 are used, our joint decoder outperforms the tree parsing systems based on 1-best trees ($+2.9$) and packed forests ($+1.1$) significantly ($p < 0.01$). This result is also better than that of using only translation features significantly (from 32.0 to 32.7, $p < 0.05$).

Not surprisingly, our decoder is much slower than pattern matching on 1-best trees and packed forests (with the same beam size). In particular, including Collins' Model 1 increases decoding time significantly because its sub-model probabilities requires back-off smoothing on the fly.

How many 1-best trees produced by our decoder are included in the parse forest produced by a standard parser? We used the Chinese parser to generate five pruned packed forests with different sizes (average number of trees stored in a forest). As shown in Table 4, only 2.22% of the trees produced by our decoder were included in the biggest forest. One possible reason is that we used sub-sentence division to reduce decoding complexity. To further investigate the matching rate, we also calculated labeled precision, which indicates how many brackets in the parse match those in the packed forest. The labeled precision on the biggest forest is 65.9%, suggesting that the 1-best trees produced by our decoder are significantly different from those in the packed forests produced by a standard parser. [8]

3.2 Parsing Evaluation

We followed Petrov and Klein (2007) to divide the Penn Chinese Treebank (CTB) version 5 as follows: Articles 1-270 and 400-1151 as the training set, Articles 301-325 as the development set, and Articles 271-300 as the test set. We used max-F_1 training (Och, 2003) to train the feature weights. We did not use sub-sentence division as the sentences in the test set have no more than 40 words.

[8] The packed forest produced by our decoder ("rule" forest) might be different from the forest produced by a monolingual parser ("parser" forest). While tree-based and forest-based decoders search in the intersection of the two forests (i.e., matched forest), our decoder directly explores the "rule" forest, which represents the true search space of tree-to-string translation. This might be the key difference of our approach from forest-based translation (Mi et al., 2008). As sub-sentence division makes direct comparison of the two forests quite difficult, we leave this to future work.

Parsing model	F_1 (%)	Time (s)
-	62.7	23.9
PCFG	65.4	24.7
Lex	79.8	48.8
PCFG + Lex	**80.6**	50.4

Table 5: Effect of parsing models on parsing performance ($\leq$ 40 words) and average decoding time (second per sentence). We use "-" to denote only using translation features.

Table 5 shows the results. Translation features were used for all configurations. Without parsing models, the F_1 score is 62.7. Adding Collins' Model 1 results in much larger gains than adding PCFG. With all parsing models integrated, our joint decoder achieves an F_1 score of 80.6 on the test set. Although lower than the F_1 score of the in-house parser that produces the noisy training data, this result is still very promising because the tree-to-string rules that construct trees in the decoding process are learned from noisy training data.

4 Related Work

Charniak et al. (2003) firstly associate lexicalized parsing model with syntax-based translation. They first run a string-to-tree decoder (Yamada and Knight, 2001) to produce an English parse forest and then use a lexicalized parsing model to select the best translation from the forest. As the parsing model operates on the target side, it actually serves as a syntax-based language model for machine translation. Recently, Shen et al. (2008) have shown that dependency language model is beneficial for capturing long-distance relations between target words. As our approach adds parsing models to the source side where the source sentence is fixed during decoding, our decoder does parse the source sentence like a monolingual parser instead of a syntax-based language model. More importantly, we integrate translation models and parsing models in a discriminative framework where they can interact with each other directly.

Our work also has connections to joint parsing (Smith and Smith, 2004; Burkett and Klein, 2008) and bilingually-constrained monolingual parsing (Huang et al., 2009a) because we use another language to resolve ambiguity for one language. However, while both joint parsing and bilingually-constrained monolingual parsing rely on the target sentence, our approach only takes a source sentence as input.

Blunsom and Osborne (2008) incorporate the source-side parse trees into their probabilistic SCFG framework and treat every source-parse PCFG rule as an individual feature. The difference is that they parse the test set before decoding so as to exploit the source syntactic information to guide translation.

More recently, Chiang (2010) has shown that ("exact") tree-to-tree translation as parsing achieves comparable performance with Hiero (Chiang, 2007) using much fewer rules. Xiao et al. (2010) integrate tokenization and translation into a single step and improve the performance of tokenization and translation significantly.

5 Conclusion

We have presented a framework for joint parsing and translation by casting tree-to-string translation as a parsing problem. While tree-to-string rules construct parse trees on the source side and translations on the target side simultaneously, parsing models can be integrated to improve both translation and parsing quality.

This work can be considered as a final step towards the continuum of tree-to-string translation: from single tree to forest and finally to the integration of parsing and translation. In the future, we plan to develop more efficient decoding algorithms, analyze forest matching systematically, and use more sophisticated parsing models.

Acknowledgement

The authors were supported by National Natural Science Foundation of China, Contracts 60736014 and 60903138, and 863 State Key Project No. 2006AA010108. We are grateful to the anonymous reviewers for their insightful comments. We thank Liang Huang, Hao Zhang, and Tong Xiao for discussions on synchronous binarization, Haitao Mi and Hao Xiong for providing and running the baseline systems, and Wenbin Jiang for helping us train parsing models.

References

Blunsom, Phil and Miles Osborne. 2008. Probabilistic inference for machine translation. In *Proc. of EMNLP 2008*.

Burkett, David and Dan Klein. 2008. Two languages are better than one (for syntactic parsing). In *Proc. of EMNLP 2008*.

Charniak, Eugene, Kevin Knight, and Kenji Yamada. 2003. Syntax-based language models for statistical machine translation. In *Proc. of MT Summit IX*.

Chiang, David. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Chiang, David. 2010. Learning to translate with source and target syntax. In *Proc. of ACL 2010*.

Collins, Michael. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.

Eisner, Jason. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proc. of ACL 2003*.

Galley, Michel, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proc. of NAACL 2004*.

Galley, Michel, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proc. of ACL 2006*.

Huang, Liang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proc. of AMTA 2006*.

Huang, Liang, Wenbin Jiang, and Qun Liu. 2009a. Bilingually-constrained (monolingual) shift-reduce parsing. In *Proc. of EMNLP 2009*.

Huang, Liang, Hao Zhang, Daniel Gildea, and Kevin Knight. 2009b. Binarization of synchronous context-free grammars. *Computational Linguistics*, 35(4):559–595.

Huang, Liang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proc. of ACL 2008*.

Liu, Yang, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proc. of ACL 2006*.

Liu, Yang, Yajuan Lü, and Qun Liu. 2009. Improving tree-to-tree translation with packed forests. In *Proc. of ACL 2009*.

Melamed, I. Dan. 2004. Statistical machine translation by parsing. In *Proc. of ACL 2004*.

Mi, Haitao, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proc. of ACL 2008*.

Och, Franz J. and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Och, Franz. 2003. Minimum error rate training in statistical machine translation. In *Proc. of ACL 2003*.

Petrov, Slav and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proc. of NAACL 2007*.

Quirk, Chris and Simon Corston-Oliver. 2006. The impact of parsing quality on syntactically-informed statistical machine translation. In *Proc. of EMNLP 2006*.

Quirk, Chris, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proc. of ACL 2005*.

Shen, Libin, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proc. of ACL 2008*.

Smith, David and Noah Smith. 2004. Bilingual parsing with factored estimation: using english to parse korean. In *Proc. of EMNLP 2004*.

Stolcke, Andreas. 2002. Srilm - an extension language model modeling toolkit. In *Proc. of ICSLP 2002*.

Xiao, Xinyan, Yang Liu, Young-Sook Hwang, Qun Liu, and Shouxun Lin. 2010. Joint tokenization and translation. In *Proc. of COLING 2010*.

Xiong, Deyi, Shuanglong Li, Qun Liu, and Shouxun Lin. 2005. Parsing the penn chinese treebank with semantic knowledge. In *Proc. of IJCNLP 2005*.

Yamada, Kenji and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proc. of ACL 2001*.

Zhang, Hao, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translatio. In *Proc. of NAACL 2007*.

Zhang, Min, Hongfei Jiang, Aiti Aw, Haizhou Li, Chew Lim Tan, and Sheng Li. 2008. A tree sequence alignment-based tree-to-tree translation model. In *Proc. of ACL 2008*.

Zhao, Bing and Yaser Al-Onaizan. 2008. Generalizing local and non-local word-reordering patterns for syntax-based machine translation. In *Proc. of EMNLP 2008*.

Semantic Role Features for Machine Translation

Ding Liu
Department of Computer Science
University of Rochester

Daniel Gildea
Department of Computer Science
University of Rochester

Abstract

We propose semantic role features for a Tree-to-String transducer to model the reordering/deletion of source-side semantic roles. These semantic features, as well as the Tree-to-String templates, are trained based on a conditional log-linear model and are shown to significantly outperform systems trained based on Max-Likelihood and EM. We also show significant improvement in sentence fluency by using the semantic role features in the log-linear model, based on manual evaluation.

1 Introduction

Syntax-based statistical machine translation (SSMT) has achieved significant progress during recent years (Galley et al., 2006; May and Knight, 2007; Liu et al., 2006; Huang et al., 2006), showing that deep linguistic knowledge, if used properly, can improve MT performance. Semantics-based SMT, as a natural extension to SSMT, has begun to receive more attention from researchers (Liu and Gildea, 2008; Wu and Fung, 2009). Semantic structures have two major advantages over syntactic structures in terms of helping machine translation. First of all, semantic roles tend to agree better between two languages than syntactic constituents (Fung et al., 2006). This property motivates the approach of using the consistency of semantic roles to select MT outputs (Wu and Fung, 2009). Secondly, the set of semantic roles of a predicate models the *skeleton* of a sentence, which is crucial to the readability of MT output. By skeleton, we mean the main structure of a sentence including the verbs and their arguments. In spite of the theoretical potential of the semantic roles, there has not been much success in using them to improve SMT systems.

Liu and Gildea (2008) proposed a semantic role based Tree-to-String (TTS) transducer by adding semantic roles to the TTS templates. Their approach did not differentiate the semantic roles of different predicates, and did not always improve the TTS transducer's performance. Wu and Fung (2009) took the output of a phrase-based SMT system Moses (Koehn et al., 2007), and kept permuting the semantic roles of the MT output until they best matched the semantic roles in the source sentence. This approach shows the positive effect of applying semantic role constraints, but it requires re-tagging semantic roles for every permuted MT output and does not scale well to longer sentences.

This paper explores ways of tightly integrating semantic role features (SRFs) into an MT system, rather than using them in post-processing or n-best re-ranking. Semantic role labeling (SRL) systems usually use sentence-wide features (Xue and Palmer, 2004; Pradhan et al., 2004; Toutanova et al., 2005); thus it is difficult to compute target-side semantic roles incrementally during decoding. Noticing that the source side semantic roles are easy to compute, we apply a compromise approach, where the target side semantic roles are generated by projecting the source side semantic roles using the word alignments between the source and target sentences. Since this approach does not perform true SRL on the target string, it cannot fully evaluate whether the source and target semantic structures are consistent. However, the approach does capture the semantic-level re-ordering of the sentences. We assume here that the MT system is capable of providing word alignment (or equivalent) information during decoding, which is generally true for current statistical MT systems.

Specifically, two types of semantic role features are proposed in this paper: a semantic role reordering feature designed to capture the skeleton-level permutation, and a semantic role deletion fea-

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 716–724,
Beijing, August 2010

ture designed to penalize missing semantic roles in the target sentence. To use these features during decoding, we need to keep track of the semantic role sequences (SRS) for partial translations, which can be generated based on the source-side semantic role sequence and the corresponding word alignments. Since the SRL system and the MT system are separate, a translation rule (e.g., a phrase pair in phrase-based SMT) could cover two partial source-side semantic roles. In such cases partial SRSs must be recorded in such a way that they can be combined later with other partial SRSs. Dealing with this problem will increase the complexity of the decoding algorithm. Fortunately, Tree-to-String transducer based MT systems (Liu et al., 2006; Huang et al., 2006) can avoid this problem by using the same syntax tree for both SRL and MT. Such an arrangement guarantees that a TTS template either covers parts of one source-side semantic role, or a few complete semantic roles. This advantage motivates us to use a TTS transducer as the MT system with which to demonstrate the use of the proposed semantic role features. Since it is hard to design a generative model to combine both the semantic role features and the TTS templates, we use a log-linear model to estimate the feature weights, by maximizing the conditional probabilities of the target strings given the source syntax trees. The log-linear model with latent variables has been discussed by Blunsom et al. (2008); we apply this technique to combine the TTS templates and the semantic role features.

The remainder of the paper is organized as follows: Section 2 describes the semantic role features proposed for machine translation; Section 3 describes how semantic role features are used and trained in a TTS transducer; Section 4 presents the experimental results; and Section 5 gives the conclusion.

2 Semantic Role Features for Machine Translation

2.1 Defining Semantic Roles

There are two semantic standards with publicly available training data: PropBank (Palmer et al., 2005) and FrameNet (Johnson et al., 2002). PropBank defines a set of semantic roles for the verbs in the Penn TreeBank using numbered roles. These roles are defined individually for each verb. For example, for the verb *disappoint*, the role name *arg1* means *experiencer*, but for the verb *wonder*, role name *arg1* means *cause*. FrameNet is motivated by the idea that a certain type of verbs can be gathered together to form a *frame*, and in the same frame, a set of semantic roles is defined and shared among the verbs. For example, the verbs *boil*, *bake*, and *steam* will be in frame *apply_heat*, and they have the semantic roles of *cook*, *food*, and *heating_instrument*. Of these two semantic standards, we choose PropBank over FrameNet for the following reasons:

1. PropBank has a simpler semantic definition than FrameNet and thus is easier for automatic labeling.

2. PropBank is built upon the Penn TreeBank and is more consistent with statistical parsers, most of which are trained on the Penn TreeBank.

3. PropBank is a larger corpus than FrameNet.

Note that the semantic standard/corpus is not crucial in this paper. Any training corpus that can be used to automatically obtain the set of semantic roles of a verb could be used in our approach.

2.2 Semantic Role Features

Ideally, we want to use features based on the true semantic roles of the MT candidates. Considering there is no efficient way of integrating SRL and MT, accurate target-side semantic roles can only be used in post-processing and re-ranking the MT outputs, where a limited number of MT candidates are considered. On the other hand, it is much easier to obtain reliable semantic roles for the source sentences. This paper uses a compromise approach, where the target-side semantic roles are projected from the source-side semantic roles using the word alignment derived from the translation process. More specifically, we define two types of semantic role features:

1. **Semantic Role Re-ordering (SRR)** This feature describes re-ordering of the source-side

semantic roles (including the predicate) in the target side. It takes the following form:

$$SrcPred : SrcRole_1, ..., SrcRole_n \Rightarrow TarRole_1, ..., TarRole_n$$

where $SrcPred$ and $SrcRole$ denotes the central verb and semantic roles in the source side, and $TarRole$ denotes the target-side roles. The source/target SRSs do not need be continuous, but there should be a one-to-one alignment between the roles in the two sides. Compared to the general re-ordering models used in statistical MT systems, this type of feature is capable of modeling skeleton-level re-ordering, which is crucial to the fluency of MT output. Because a predicate can have different semantic role sequences in different voices, *passive/active* are tagged for each occurrence of the verbs based on their POS and preceding words. Figure 1 shows examples of the feature SRR.

2. **Deleted Roles (DR)** are the individual source-side semantic roles which are deleted in the MT outputs, taking the form of:

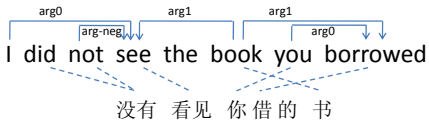
$$SrcPred : SrcRole \Rightarrow deleted$$

DR is meant to penalize the deletion of the semantic roles. Though most statistical MT systems have penalties for word deletion, it is still useful to make separate features for the deletion of semantic roles, which is considered more harmful than the deletion of non-core components (e.g., modifiers) and deserves more serious penalty. Examples of the deletion features can be found in Figure 1.

Both types of features can be made non-lexicalized by removing the actual verb but retaining its voice information in the features. Non-lexicalized features are used in the system to alleviate the problem of sparse verbs.

3 Using Semantic Role Features in Machine Translation

This section describes how to use the proposed semantic role features in a Tree-to-String transducer,



SRR:
see-active: arg-neg verb ➔ arg-neg verb
borrowed-active: arg1 arg0 ➔ arg0 arg1
borrowed-active: arg1 verb ➔ verb arg1
borrowed-active: arg0 verb ➔ arg0 verb
borrowed-active: arg1 arg0 verb ➔ arg0 verb arg1
DR:
see-active: arg0 ➔ deleted

Figure 1: Examples of the semantic role features

assuming that the semantic roles have been tagged for the source sentences. We first briefly describe the basic Tree-to-String translation model used in our experiments, and then describe how to modify it to incorporate the semantic role features.

3.1 Basic Tree-to-String Transducer

A Tree-to-String transducer receives a syntax tree as its input and, by recursively applying TTS templates, generates the target string. A TTS template is composed of a left-hand side (LHS) and a right-hand side (RHS), where the LHS is a subtree pattern and the RHS is a sequence of variables and translated words. The variables in the RHS of a template correspond to the bottom level non-terminals in the LHS's subtree pattern, and their relative order indicates the permutation desired at the point where the template is applied to translate one language to another. The variables are further transformed, and the recursive process goes on until there are no variables left. The formal description of a TTS transducer is given by Graehl and Knight (2004), and our baseline approach follows the *Extended Tree-to-String Transducer* defined by Huang et al. (2006). For a given derivation (decomposition into templates) of a syntax tree, the translation probability is computed as the product of the templates which generate both the source syntax trees and the target translations.

$$Pr(S \mid T, D^*) = \prod_{t \in D^*} Pr(t)$$

Here, S denotes the target sentence, T denotes the source syntax tree, and D^* denotes the derivation of T. In addition to the translation model, the

function DECODE(T)
  for tree node v of T in bottom-up order **do**
    for template t applicable at v **do**

$\{c_1, c_2\}$=match(v, t);
$s.leftw = c_1.leftw$;
$s.rightw = c_2.rightw$;
$s.val = c_1.val \times c_2.val$;
$s.val \times= Pr(t)$;
$s.val \times= Pr(c_2.leftw | c_1.rightw)$;
add s to v's beam;

Figure 2: Decoding algorithm for the standard Tree-to-String transducer. $leftw$/$rightw$ denote the left/right boundary word of s. c_1, c_2 denote the descendants of v, ordered based on RHS of t.

TTS system includes a trigram language model, a deletion penalty, and an insertion bonus. The bottom-up decoding algorithm for the TTS transducer is sketched in Figure 2. To incorporate the n-gram language model, states in the algorithm denote a tree node's best translations with different left and right boundary words. We use standard beam-pruning to narrow the search space. To simplify the description, we assume in Figure 2 that a bigram language model is used and all the TTS templates are binarized. It is straightforward to generalize the algorithm for larger n-gram models and TTS templates with any number of children in the bottom using target-side binarized combination (Huang et al., 2006).

3.2 Modified Tree-to-String Transducer with Semantic Role Features

Semantic role features can be used as an auxiliary translation model in the TTS transducer, which focuses more on the skeleton-level permutation. The model score, depending on not only the input source tree and the derivation of the tree, but also the semantic roles of the source tree, can be formulated as:

$$Pr(S \mid T, D^*) = \prod_{f \in F(S, T.role, D^*)} Pr(f)$$

where T denotes the source syntax tree with semantic roles, $T.role$ denotes the semantic role sequence in the source side and $F(S.role, T.role, D^*)$ denotes the set of defined semantic role features over $T.role$ and the target side semantic role sequence $S.role$. Note that given $T.role$ and the derivation D^*, $S.role$ can

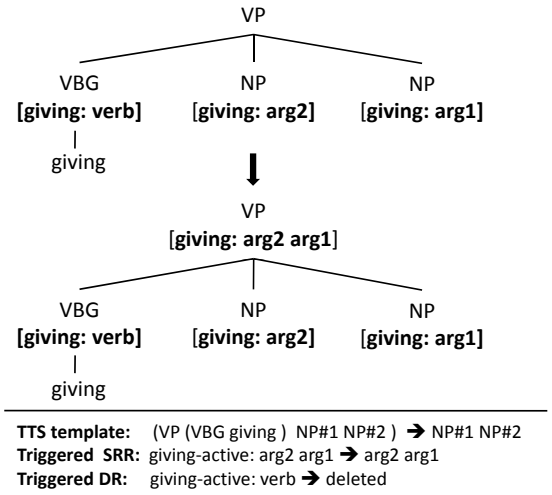


Figure 3: An example showing the combination of the semantic role sequences of the states. Above/middle is the state information before/after applying the TTS template, and bottom is the used TTS template and the triggered SRFs during the combination.

be easily derived. Now we show how to incorporate the two types of semantic role features into a TTS transducer. To use the semantic role re-ordering feature SRR, the states in the decoding algorithm need to be expanded to encode the target-side SRSs. The SRSs are initially attached to the translation states of the source tree con-

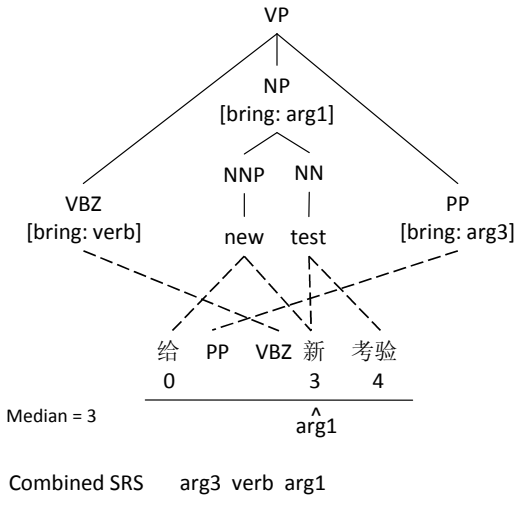


Figure 4: An example showing how to compute the target side position of a semantic role by using the median of its aligning points.

stituents which are labeled as semantic roles for some predicate. These semantic roles are then accumulated with re-ordering and deletion operations specified by the TTS templates as the decoding process goes bottom-up. Figure 5 shows the decoding algorithm incorporating the SRR features. The model component corresponding to the feature SRR is computed when combining two translation states. I.e., the probabilities of the SRR features composed based on the semantic roles of the two combining states will be added into the combined state. See Figure 3 for examples. The theoretical upper bound of the decoding complexity is $O(NM^{4(n-1)}R(\sum_{i=0}^{C}\frac{C!}{i!})^V)$, where N is the number of nodes in the source syntax tree, M is the vocabulary size of the target language, n is the order of the n-gram language model, R is the maximum number of TTS templates which can be matched at a tree node, C is the maximum number of roles of a verb, and V is the maximum number of verbs in a sentence. In this formula, $\sum_{i=0}^{C}\frac{C!}{i!}$ is the number of role sequences obtained by first choosing i out of C possible roles and then permuting the i roles. This theoretical upper bound is not reached in practice, because the number of possible TTS templates applicable at a tree node is very limited. Furthermore, since we apply beam pruning at each tree node, the running time is controlled by the beam size, and is linear in the size of the tree.

The re-ordering of the semantic roles from source to target is computed for each TTS template as part of the template extraction process, using the word-level alignments between the LHS/RHS of the TTS template (e.g., Figure 3). This is usually straightforward, with the exception of the case where the words that are aligned to a particular role's span in the source side are not continuous in the target side, as shown in Figure 4. Since we are primarily interested in the relative order of the semantic roles, we approximate each semantic role's target side position by the median of the word positions that is aligned to. If more than one semantic role is mapped to the same position in the target side, their source side order will be used as their target side order, i.e., monotonic translation is assumed for those semantic roles. Figure 4 shows an example of calculating the target side

function DECODE(T)
  for tree node v of T in bottom-up order **do**
    for template t applicable at v **do**
      $\{c_1, c_2\}$=match(v, t);
      $s.leftw$ = $c_1.leftw$;
      $s.rightw$ = $c_2.rightw$;
      $s.role$ = concatenate($c_1.role$, $c_2.role$);
      if v is a semantic role **then**
        set *s.role* to *v.role*;
      $s.val$ = $c_1.val \times c_2.val$;
      $s.val \times= Pr(t)$;
      $s.val \times= Pr(c_2.leftw|c_1.rightw)$;
    ▷ Compute the probabilities associated with semantic roles
      $s.val \times= \prod_{f \in Sema(c_1.role, c_2.role, t)} Pr(f)$;
      add s to v's beam;

Figure 5: Decoding algorithm using semantic role features. $Sema(c_1.role, c_2.role, t)$ denotes the triggered semantic role features when combining two children states, and examples can be found in Figure 3.

SRS based on a complicated TTS template. The word alignments in the TTS templates are also used to compute the deletion feature DR. Whenever a semantic role is deleted in a TTS template's RHS, the corresponding deletion penalty will be applied.

3.3 Training

We describe two alternative methods for training the weights for the model's features, including both the individual TTS templates and the semantic role features. The first method maximizes data likelihood as is standard in EM, while the second method maximizes conditional likelihood for a log-linear model following Blunsom et al. (2008).

3.3.1 Maximizing Data Likelihood

The standard way to train a TTS translation model is to extract the minimum TTS templates using GHKM (Galley et al., 2004), and then normalize the frequency of the extracted TTS templates (Galley et al., 2004; Galley et al., 2006; Liu et al., 2006; Huang et al., 2006). The probability of the semantic features SRR and DR can be computed similarly, given that SRR and DR can be derived from the paired source/target sentences and the word alignments between them. We refer to this model as max-likelihood training and normalize the counts of TTS templates and semantic features based on their roots and predicates respectively.

We wish to overcome noisy alignments from GIZA++ and learn better TTS rule probabilities by re-aligning the data using EM within the TTS

E-step:

for all pair of syntax tree T and target string S **do**
  for all TTS Template t, semantic features f **do**
    $EC(t)$ += $\frac{\sum_{D:t\in D} Pr(S,T,D)}{\sum_{D'} Pr(S,T,D')}$;
    $EC(f)$ += $\frac{\sum_{D:f\in D} Pr(S,T,D)}{\sum_{D'} Pr(S,T,D')}$;

M-step:

for all TTS Template t, semantic features f **do**
  $Pr(t) = \frac{EC(t)}{\sum_{t':t'.root=t.root} EC(t')}$;
  $Pr(f) = \frac{EC(f)}{\sum_{f':f'.predicate=t.predicate} EC(f')}$;

Figure 6: EM Algorithm For Estimating TTS Templates and Semantic Features

function COMPUTE_PARTITION(T)
  for tree node v of T in bottom-up order **do**
    for template t applicable at v **do**
      for $\{s_1, s_2\}$=Match(v, t) **do**
        $s.sum$ += $s_1.sum \times s_2.sum \times exp(\lambda_t + \sum_{f\in Sema(s_1,s_2,t)} \lambda_f)$;
        $s.role$ = concatenate($s_1.role$, $s_2.role$);
        add s to v;
  for state s in $root$ **do** res += $s.sum$;
  return res;

Figure 7: Computing the partition function of the conditional probability $Pr(S|T)$. $Sema(s_1, s_2, t)$ denotes all the semantic role features generated by combining s_1 and s_2 using t.

framework (May and Knight, 2007). We can estimate the expected counts of the TTS templates and the semantic features by formulating the probability of a pair of source tree and target string as:

$$\sum_D Pr(S,T,D) = \sum_D \left(\prod_{t\in D} Pr(t) \prod_{f\in F(S,T.role,D)} Pr(f) \right)$$

Though the above formulation, which makes the total probability of all the pairs of trees and strings less than 1, is not a strict generative model, we can still use the EM algorithm (Dempster et al., 1977) to estimate the probability of the TTS templates and the semantic features, as shown in Figure 6.

The difficult part of the EM algorithm is the *E-step*, which computes the expected counts of the TTS templates and the semantic features by summing over all possible derivations of the source trees and target strings. The standard inside-outside algorithm (Graehl and Knight, 2004) can be used to compute the expected counts of the TTS templates. Similar to the modification made in the TTS decoder, we can add the target-side semantic role sequence to the dynamic programming states of the inside-outside algorithm to compute the expected counts of the semantic features. This way each state (associated with a source tree node) represents a target side span and the partial SRSs. To speed up the training, a beam is created for each target span and only the top rated SRSs in the beam are kept.

3.3.2 Maximizing Conditional Likelihood

A log-linear model is another way to combine the TTS templates and the semantic features together. Considering that the way the semantic role features are defined makes it impossible to design a sound generative model to incorporate these features, a log-linear model is also a theoretically better choice than the EM algorithm. If we directly translate the EM algorithm into the log-linear model, the problem becomes maximizing the data likelihood represented by feature weights instead of feature probabilities:

$$Pr(S,T) = \frac{\sum_D \exp \sum_i \lambda_i f_i(S,T,D)}{\sum_{S',T'} \sum_{D'} \exp \sum_i \lambda_i f_i(S',T',D')}$$

where the features f include both the TTS templates and the semantic role features. The numerator in the formula above can be computed using the same dynamic programming algorithm used to compute the expected counts in the EM algorithm. However, the partition function (denominator) requires summing over all possible source trees and target strings, and is infeasible to compute. Instead of approximating the partition function using methods such as sampling, we change the objective function from the data likelihood to the conditional likelihood:

$$Pr(S \mid T) = \frac{\sum_D \exp \sum_i \lambda_i f_i(S,T,D)}{\sum_{S'\in all(T)} \sum_{D'} \exp \sum_i \lambda_i f_i(S',T,D')}$$

where $all(T)$ denotes all the possible target strings which can be generated from the source tree T. Given a set of TTS templates, the new partition function can be efficiently computed using the dynamic programming algorithm shown in Figure 7. Again, to simplify the illustration, only binary TTS templates are used. Using the conditional probability as the objective function not only reduces the computational cost, but also corresponds better to the TTS decoder, where the best MT output is

selected only among the possible candidates which can be generated from the input source tree using TTS templates.

The derivative of the logarithm of the objective function (over the entire training corpus) w.r.t. a feature weight can be computed as:

$$\frac{\partial \log \prod_{S,T} Pr(S \mid T)}{\partial \lambda_i} = \sum_{S,T} \{EC_{D|S,T}(f_i) - EC_{S'|T}(f_i)\}$$

where $EC_{D|S,T}(f_i)$, the expected count of a feature over all derivations given a pair of tree and string, can be computed using the modified inside-outside algorithm described in Section 3.2, and $EC_{S'|T}(f_i)$, the expected count of a feature over all possible target strings given the source tree, can be computed in a similar way to the partition function described in Figure 7. With the objective function and its derivatives, a variety of optimization methods can be used to obtain the best feature weights; we use LBFGS (Zhu et al., 1994) in our experiments. To prevent the model from overfitting the training data, a weighted Gaussian prior is used with the objective function. The variance of the Gaussian prior is tuned based on the development set.

4 Experiments

We train an English-to-Chinese translation system using the FBIS corpus, where 73,597 sentence pairs are selected as the training data, and 500 sentence pairs with no more than 25 words on the Chinese side are selected for both the development and test data.[1] Charniak (2000)'s parser, trained on the Penn Treebank, is used to generate the English syntax trees. To compute the semantic roles for the source trees, we use an in-house max-ent classifier with features following Xue and Palmer (2004) and Pradhan et al. (2004). The semantic role labeler is trained and tuned based on sections 2–21 and section 24 of PropBank respectively. The standard role-based F-score of our semantic role labeler is 88.70%. Modified Kneser-Ney trigram models are trained using SRILM (Stolcke, 2002) on the Chinese portion of the training data. The model (n-gram language model, TTS templates, SRR, DR) weights of the transducer are tuned based on the development set using a grid-based line search, and the translation results are evaluated based on a single Chinese reference using BLEU-4 (Papineni et al., 2002). Huang et al. (2006) used character-based BLEU as a way of normalizing inconsistent Chinese word segmentation, but we avoid this problem as the training, development, and test data are from the same source.

The baseline system in our experiments uses the TTS templates generated by using GHKM and the union of the two single-direction alignments generated by GIZA++. Unioning the two single-direction alignments yields better performance for the SSMT systems using TTS templates (Fossum et al., 2008) than the two single-direction alignments and the heuristic diagonal combination (Koehn et al., 2003). The two single-direction word alignments as well as the union are used to generate the initial TTS template set for both the EM algorithm and the log-linear model. The initial TTS templates' probabilities/weights are set to their normalized counts based on the root of the TTS template (Galley et al., 2006). To test semantic role features, their initial weights are set to their normalized counts for the EM algorithm and to 0 for the log-linear model. The performance of these systems is shown in Table 1. We can see that the EM algorithm, based only on TTS templates, is slightly better than the baseline system. Adding semantic role features to the EM algorithm actually hurts the performance, which is not surprising since the combination of the TTS templates and semantic role features does not yield a sound generative model. The log-linear model based on TTS templates achieves significantly better results than both the baseline system and the EM algorithm. Both improvements are significant at $p < 0.05$ based on 2000 iterations of paired bootstrap resampling of the test set (Koehn, 2004).

Adding semantic role features to the log-linear model further improves the BLEU score. One problem in our approach is the sparseness of the verbs, which makes it difficult for the log-linear model to tune the lexicalized semantic role features. One way to alleviate this problem is to make features based on verb classes. We first tried using the verb

[1] The total 74,597 sentence pairs used in experiments are those in the FBIS corpus whose English part can be parsed using Charniak (2000)'s parser.

	TTS Templates	+ SRF	+ Verb Class
Union	15.6	–	–
EM	15.9	15.5	15.6
Log-linear	17.1	17.4	17.6

Table 1: BLEU-4 scores of different systems

	equal	better	worse
With SRF vs. W/O SRF	72%	20.2%	7.8%

Table 2: Distribution of the sentences where the semantic role features give no/positive/negative impact to the sentence fluency in terms of the completeness and ordering of the semantic roles.

classes in VerbNet (Dang et al., 1998). Unfortunately, VerbNet only covers about 34% of the verb tokens in our training corpus, and does not improve the system's performance. We then resorted to automatic clustering based on the aspect model (Hofmann, 1999; Rooth et al., 1999). The training corpus used in clustering is the English portion of the selected FBIS corpus. Though automatically obtained verb clusters lead to further improvement in BLEU score, the total improvement from the semantic role features is not statistically significant. Because BLEU-4 is biased towards the adequacy of the MT outputs and may not effectively evaluate their fluency, it is desirable to give a more accurate evaluation of the sentence's fluency, which is the property that semantic role features are supposed to improve. To do this, we manually compare the outputs of the two log-linear models with and without the semantic role features. Our evaluation focuses on the completeness and ordering of the semantic roles, and *better*, *equal*, *worse* are tagged for each pair of MT outputs indicating the impact of the semantic role features. Table 2 shows the manual evaluation results based on the entire test set, and the improvement from SRF is significant at $p < 0.005$ based on a t-test. To illustrate how SRF impacts the translation results, Figure 8 gives 3 examples of the MT outputs with and without the SRFs.

Source	Launching$_1$ New$_2$ Diplomatic$_3$ Offensive$_4$
SRF On	实施$_1$ 新的$_2$ 外交$_3$ 攻势$_4$
SRF Off	新的$_2$ 外交$_3$ 攻势$_4$

Source	It$_1$ is$_2$ therefore$_3$ necessary$_4$ to$_5$ speed$_6$ up$_7$ the$_8$ transformation$_9$ of$_{10}$ traditional$_{11}$ industries$_{12}$ with$_{13}$ high$_{14}$ technologies$_{15}$
SRF On	所以$_{123}$ 要$_4$ 加快$_{6,7}$ 高新技术$_{14,15}$ 改造$_9$ 传统产业$_{11,12}$
SRF Off	所以$_{123}$ 要$_4$ 高技术$_{14,15}$ ，加快$_{6,7}$ 传统产业$_{11,12}$ 改造$_9$

Source	A$_1$ gratifying$_2$ change$_3$ also$_4$ occurred$_5$ in$_6$ the$_7$ structure$_8$ of$_9$ ethnic$_{10}$ minority$_{11}$ cadres$_{12}$
SRF On	少数民族$_{10,11}$ 结构$_8$ 也$_4$ 发生$_5$ 可喜的$_2$ 变化$_3$
SRF Off	一个$_1$ 可喜的$_2$ 变化$_3$, 还在$_4$ 少数民族$_{10,11}$ 干部的 结构$_8$

Figure 8: Examples of the MT outputs with and without SRFs. The first and second example shows that SRFs improve the completeness and the ordering of the MT outputs respectively, the third example shows that SRFs improve both properties. The subscripts of each Chinese phrase show their aligned words in English.

5 Conclusion

This paper proposes two types of semantic role features for a Tree-to-String transducer: one models the reordering of the source-side semantic role sequence, and the other penalizes the deletion of a source-side semantic role. These semantic features and the Tree-to-String templates, trained based on a conditional log-linear model, are shown to significantly improve a basic TTS transducer's performance in terms of BLEU-4. To avoid BLEU's bias towards the adequacy of the MT outputs, manual evaluation is conducted for sentence fluency and significant improvement is shown by using the semantic role features in the log-linear model. Considering our semantic features are the most basic ones, using more sophisticated features (e.g., the head words and their translations of the source-side semantic roles) provides a possible direction for further experimentation.

Acknowledgments This work was funded by NSF IIS-0546554 and IIS-0910611.

References

Blunsom, Phil, Trevor Cohn, and Miles Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL-08)*, Columbus, Ohio.

Charniak, Eugene. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL-01*, pages 132–139.

Dang, Hoa Trang, Karin Kipper, Martha Palmer, and

Joseph Rosenzweig. 1998. Investigating regular sense extensions based on intersective Levin classes. In *COLING/ACL-98*, pages 293–299, Montreal. ACL.

Dempster, A. P., N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the *EM* algorithm. *Journal of the Royal Statistical Society*, 39(1):1–21.

Fossum, Victoria, Kevin Knight, and Steven Abney. 2008. Using syntax to improveword alignment precision for syntax-based machine translation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, Columbus, Ohio. ACL.

Fung, Pascale, Zhaojun Wu, Yongsheng Yang, and Dekai Wu. 2006. Learning of Chinese/English semantic structure mapping. In *IEEE/ACL 2006 Workshop on Spoken Language Technology*, Aruba.

Galley, Michel, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of NAACL-04*, pages 273–280.

Galley, Michel, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of COLING/ACL-06*, pages 961–968, July.

Graehl, Jonathan and Kevin Knight. 2004. Training tree transducers. In *Proceedings of NAACL-04*.

Hofmann, Thomas. 1999. Probabilistic latent semantic analysis. In *Uncertainity in Artificial Intelligence, UAI'99*, Stockholm.

Huang, Liang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proceedings of the 7th Biennial Conference of the Association for Machine Translation in the Americas (AMTA)*, Boston, MA.

Johnson, Christopher R., Charles J. Fillmore, Miriam R. L. Petruck, Collin F. Baker, Michael Ellsworth, Josef Ruppenhofer, and Esther J. Wood. 2002. FrameNet: Theory and practice. Version 1.0, http://www.icsi.berkeley.edu/framenet/.

Koehn, Philipp, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of NAACL-03*, Edmonton, Alberta.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL, Demonstration Session*, pages 177–180.

Koehn, Philipp. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP*, pages 388–395, Barcelona, Spain, July.

Liu, Ding and Daniel Gildea. 2008. Improved tree-to-string transducers for machine translation. In *ACL Workshop on Statistical Machine Translation (ACL08-SMT)*, pages 62–69, Columbus, Ohio.

Liu, Yang, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of COLING/ACL-06*, Sydney, Australia, July.

May, Jonathan and Kevin Knight. 2007. Syntactic re-alignment models for machine translation. In *Proceedings of EMNLP*.

Palmer, Martha, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of ACL-02*.

Pradhan, Sameer, Wayne Ward, Kadri Hacioglu, James Martin, , and Dan Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *Proceedings of NAACL-04*.

Rooth, Mats, Stefan Riezler, Detlef Prescher, Glenn Carroll, and Franz Beil. 1999. Inducing a semantically annotated lexicon via EM-based clustering. In *Proceedings of the 37th Annual Meeting of the ACL*, pages 104–111, College Park, Maryland.

Stolcke, Andreas. 2002. SRILM - an extensible language modeling toolkit. In *International Conference on Spoken Language Processing*, volume 2, pages 901–904.

Toutanova, Kristina, Aria Haghighi, and Christopher Manning. 2005. Joint learning improves semantic role labeling. In *Proceedings of ACL-05*, pages 589–596.

Wu, Dekai and Pascale Fung. 2009. Semantic roles for smt: A hybrid two-pass model. In *Proceedings of the HLT-NAACL 2009: Short Papers*, Boulder, Colorado.

Xue, Nianwen and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of EMNLP*.

Zhu, Ciyou, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. 1994. L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization. Technical report, ACM Trans. Math. Software.

TimeML Events Recognition and Classification: Learning CRF Models with Semantic Roles

Hector Llorens, Estela Saquete, Borja Navarro-Colorado
Natural Language Processing Group
University of Alicante
{hllorens,stela,borja}@dlsi.ua.es

Abstract

This paper analyzes the contribution of semantic roles to TimeML event recognition and classification. For that purpose, an approach using conditional random fields with a variety of morphosyntactic features plus semantic roles features is developed and evaluated. Our system achieves an F1 of 81.4% in recognition and a 64.2% in classification. We demonstrate that the application of semantic roles improves the performance of the presented system, especially for nominal events.

1 Introduction

Event recognition and classification has been pointed out to be very important to improve complex natural language processing (NLP) applications such as automatic summarization (Daniel et al., 2003) and question answering (QA) (Pustejovsky, 2002). Natural language (NL) texts often describe sequences of events in a time line. In the context of summarization, extracting such events may aid in obtaining better summaries when these have to be focused on specific happenings. In the same manner, the access to such information is crucial for QA systems attempting to address questions about events.

The analysis of events as well as the classification of the different forms they adopt in NL text is not a new issue (Vendler, 1967). It relates not only to linguistics but different scientific areas such as philosophy, psychology, etc.

In NLP, different definitions of event can be found regarding the target application.

On the one hand, in topic detection and tracking (Allan, 2002), event is defined as an instance of a topic identified at document level describing something that happen (e.g., "wars"). The aim of this task is to cluster documents on the same topic, that is to say, the same event.

On the other hand, information extraction (IE) provides finer granularity event definitions. IE proposes standard schemes to annotate the individual events within the scope of a document. STAG scheme (2000) was aimed to identify events in news and their relationship with points in a temporal line. More recently, TimeML (Pustejovsky et al., 2003a) presented a rich specification for annotating events in NL text extending the features of the previous one.

This paper is focused on the TimeML view of events. TimeML defines events as situations that *happen or occur*, or elements describing *states or circumstances* in which something obtains or holds the truth. These events are generally expressed by tensed or untensed verbs, nominalizations, adjectives, predicative clauses or prepositional phrases. TimeML guidelines define seven classes of events:

- *Reporting*. Action of a person or organization declaring or narrating an event (e.g., "say")
- *Perception*. Physical perception of another event (e.g., "see", "hear")
- *Aspectual*. Aspectual predication of another event (e.g., "start", "continue")
- *I_Action*. Intensional action (e.g., "try")
- *I_State*. Intensional state (e.g., "feel", "hope")
- *State*. Circumstance in which something holds the truth (e.g., "war", "in danger")
- *Occurrence*. Events that describe things that happen (e.g., "erupt", "arrive")

The following sentence shows an example of an occurrence event and a state event.

```
It's <EVENT class="OCCURRENCE">turning</EVENT>
out to be another <EVENT class="STATE">bad</EVENT>
financial week.
```


Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 725–733,
Beijing, August 2010

The automatic annotation of events has been addressed with different data-driven approaches. Current approaches are mainly based on morphosyntactic information. Our hypothesis is that semantic roles, as higher language level analysis information, may be useful as additional feature to improve the performance of such approaches.

Within this setting, the main objective of this paper is to analyze (1) the contribution of semantic roles, as additional feature, and (2) the influence of conditional random fields (CRFs), as machine learning (ML) technique, in the events automatic recognition and classification task.

This paper is structured as follows. Firstly, related work in the task is reviewed in Section 2. The next section provides a detailed description of our proposal to address event recognition and classification. After that, Section 4 includes an evaluation of the proposal, and a comparative analysis of the results. Finally, conclusions are drawn in Section 5.

2 Related Work

There is only one corpus available annotated with TimeML events: TimeBank (Pustejovsky et al., 2003b). Hence, all the approaches regarding TimeML events extraction have been evaluated using this corpus.

EVITA system (Saurí et al., 2005) recognizes events by combining linguistic and statistical techniques. The main features used to manually encode event recognition rules are the following: part-of-speech (PoS) tagging, lemmatizing, chunking, lexical lookup and contextual parsing. Furthermore, WordNet information combined with Bayesian learned disambiguation was used to identify nominal events. EVITA obtained 74.03% precision, 87.31% recall, and 80.12% $F_{\beta=1}$ in event recognition over TimeBank.

Boguraev and Ando (2005) present an evaluation on automatic TimeML events annotation. They set out the task as a classification problem and used a robust risk minimization (RRM) classifier to solve it. The $F_{\beta=1}$ results obtained by a 5-fold cross validation over TimeBank were 78.6% for recognition and 61.3% for classification. Moreover, they evaluated the impact of applying word-profiling techniques over their approach to exploit unannotated data. Using this additional information, the $F_{\beta=1}$ results improved to 80.3% and 64.0%. In this evaluation, neither precision nor recall were given.

STEP (Bethard and Martin, 2006) is a system for TimeML event recognition and classification. This approach uses a rich set of textual, morphological, dependency and WordNet hypernymy features to build a Support Vector Machine (SVM) model. The model was trained using 9/10 of the TimeBank. The test, carried out using the remaining 1/10 of the corpus, obtained a 82.0% precision, 70.6% recall and 75.9% $F_{\beta=1}$ for recognition and a 66.7% precision, 51.2% recall and 57.9% $F_{\beta=1}$ for classification.

Finally, March and Baldwin (2008) present an evaluation on event recognition using a multiclass classifier (BSVM). The main features used to train the classifier are word and PoS context window, stop words removal and feature generalization through words grouping (numbers, named entities, etc.). The result for the best feature combination in a 10-fold cross validation over TimeBank was 76.4% $F_{\beta=1}$.

It is worth mentioning that there are two versions of the TimeBank corpus, 1.1 and 1.2. The latest version is the current gold standard. Both versions consist of the same documents[1], mainly news articles and transcribed broadcast news from different domains. EVITA is the only reference which used TimeBank 1.2 while the rest of reviewed references used TimeBank 1.1.

3 Our proposal: semantic roles enhancing a CRF model

In this section, the motivation for our proposal, and our specific approach are presented.

3.1 Motivation

The next two subsections describe the main feature (semantic roles) and the ML algorithm (CRFs) we selected to address event recognition and classification; and the reasons why we think they could be useful in that task.

[1] Except 3 documents removed in TimeBank 1.2

3.1.1 Semantic roles

Semantic role labeling (SRL) has achieved important results in the last years (Gildea and Jurafsky, 2002). For each predicate in a sentence, semantic roles identify all constituents, determining their arguments (agent, patient, etc.) and their adjuncts (locative, temporal, etc.). Currently, there exist different role sets aimed to cover opposed requirements. They range from more specific, such as FrameNet (Baker et al., 1998), to more general like PropBank (Palmer et al., 2005). Figure 1 illustrates a semantic role labeled sentence.

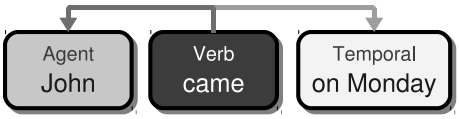


Figure 1: Semantic roles example

Many research efforts into the application of semantic roles demonstrated that this information is useful for different NLP purposes (Melli et al., 2006). Focusing on TimeML, semantic roles have been applied to temporal expressions recognition (Llorens et al., 2009), and temporal links classification (Hagège and Tannier, 2007). However, they have not been used to recognize and classify TimeML events.

Semantic roles provide structural relations of the predicates in which events may participate. Beyond syntactic relations expressed by means of the different types of phrases, semantic roles give further information about semantic relations between the arguments of a predicate. Therefore, as richer information, roles may better distinguish tokens to be candidate events. In addition, different semantic role settings may represent specific event classes.

Example 1 shows four sentences annotated with PropBank semantic roles (in square brackets) in which the noun "control" participates. In the sentences 1 and 2, "control" does not represent an event, while in the sentences 3 and 4, it represents an *state* event. It can be seen that the noun "control", when it is contained by A1 role it may represent an event. However, it is not an event when contained by A0 or AM-MNR roles. The analysis may also take into account the governing verb. In the example, we could specify that "control" represents an event when contained by A1 role of "seek" and "obtain" verbs; and the opposite for the A0 role of "emerge" and the AM-MNR of "had".

(1) 1. "[Control procedures A0] will emerge"
2. "[Iraq A0] had [thousands of Americans A1] [under its control AM-MNR]"
3. "[Crane Co. A0] may obtain [control of Milton Roy Corp. A1]"
4. "[Pattison's A0] decided to seek [control A1]"

Our hypothesis is that semantic roles, as additional information, may help in the recognition and classification of events. The information about the role of a token and the verb it depends on, or the set of roles of the sentence, could be useful for determining whether a token or a sequence of tokens is an event or not. Due to the fact that roles represent high level information in NL text, they are more independent from word tokens. Hence, roles may aid in learning more general models that could improve the results of approaches focused on lower level information.

3.1.2 CRF probabilistic model

Conditional Random Fields is a popular and efficient ML technique for supervised sequence labeling (Lafferty et al., 2001). CRFs are undirected graphical models, a special case of conditionally-trained finite state machines. A key advantage of CRFs is their flexibility to include a wide variety of arbitrary, non-independent features of the input.

We see the task set out in this paper as a sequence labeling problem. Assume X is a random variable over data sequences to be labeled, and Y is a random variable over the corresponding label sequences (hidden), being all Y components (Y_i) members of a finite label alphabet γ. X might range over NL sentences and Y range over event annotations of those sentences, with γ the set of possible event IOB2[2] labels. The following example illustrates the event recognition problem.

(2)

X	Y	
was	?	
another	?	B-EVENT
bad	?	? = I-EVENT
week	?	O

[2]IOB2 format: (B)egin, (I)nside, and (O)utside

The variables X and Y are jointly distributed over both label and observation sequences. However, unlike Hidden Markov Models (generative) in which $p(X, Y)$, CRFs (discriminative) construct a conditional model from paired observation and label sequences: $p(Y|X)$. Graphically, CRFs are represented by undirected graphs, $G = (V, E)$ such that $Y = (Y_v),\ v\ \epsilon\ V$, so that Y is indexed by the vertices of G. Then (X, Y) is a *conditional random field* if Y_v variables obey the *Markov property* with respect to the graph when conditioned on X:

$$P(Y_v|X, Y_w, v \neq w) = P(Y_v|X, Y_w, v \sim w),$$

where $v \sim w$ means that Y_v and Y_w are connected neighbors in G.

To extend the problem to event classification, the alphabet γ must be extended with the event classes (state, aspectual, etc.).

CRFs have been successfully applied to many sequence labeling tasks (Sha and Pereira, 2003; McCallum and Li, 2003).

From our point of view, the task addressed in this paper is well suited for this ML technique. Events may depend on structural properties of NL sentences. Not only the word sequence, but morphological, syntactic and semantic information is related with the event structure (Tenny and Pustejovsky, 2000).

For example, sequences of verbs may represent *i_action+occurrence* or *aspectual+occurrence* events (see Example 3).

(3) "The president will <EVENT class="i_action"> try </EVENT> to <EVENT class="occurrence"> assist </EVENT> to the <EVENT class="occurrence"> conference </EVENT>"

"Saddam will <EVENT class="aspectual"> begin </EVENT> <EVENT class="occurrence"> withdrawing </EVENT> troops from Iranian territory on Friday"

In addition, for instance, many *state* event instances are represented by "to be" plus a variable quality (see Example 4).

(4) "It is <EVENT class="occurrence"> turning </EVENT> out to be another <EVENT class="state"> bad </EVENT> financial week."

Given this analysis, our hypothesis is that CRFs will be useful in the recognition of events in which the sequential and structural properties are relevant.

3.2 Approach description

This paper proposes CRFs as learning method to infer an event recognition and classification model. Our system includes CRF++ toolkit[3] for training and testing our approach. The learning process was done using *CRF-L2* algorithm and hyper-parameter C=1.

The definition of the features is crucial for the architecture of the system. The features used in our approach are grouped in two feature sets. On the one hand, general features, which comprise morphosyntactic and ontological information. On the other hand, semantic roles features, which are the main focus of this paper.

The general features used to train our CRF model are described regarding each language analysis level.

- **Morphological**: The lemma and PoS context, in a 5-window (-2,+2), was employed. This basic linguistic feature showed good results in different NLP tasks, as well as in event recognition and classification (March and Baldwin, 2008). Tokenization, PoS and lemmatization were obtained using TreeTagger (Schmid, 1994).

- **Syntactic**: Different events are contained in particular types of phrases and syntactic dependencies. This feature tries to tackle this by considering syntactic information. Charniak parser (Charniak and Johnson, 2005) was used to obtain the syntactic tree.

- **Lexical semantics**: WordNet (Fellbaum, 1998) top ontology classes have been widely used to represent word meaning at ontological level, and demonstrated its worth in many tasks. We obtained the four top classes for each word.

The specific semantic roles features used to enhance the training framework of the CRF model were developed considering PropBank role set. PropBank was applied in our system due to the high coverage it offers in contrast to FrameNet. In order to get PropBank semantic roles, the CCG

[3] http://crfpp.sourceforge.net/

SRL tool (Punyakanok et al., 2004) was used for labeling the corpus.

- **Role**: For each token, we considered the role regarding the verb the token depends on. Semantic roles information may be useful for distinguish particular lemmas that are events only when appearing under a precise role.
- **Governing verb**: The verb to which the current token holds a particular role. This may distinguish tokens appearing under the influence of different verbs.
- **Role+verb combination**: The previous two features were combined to capture the relation between them. This introduces new classification information by distinguishing roles depending on different verbs. The importance of this falls especially on the numbered roles of PropBank (A0, A1, ...) holding different meanings when depending on different verbs.
- **Role configuration**: This consists of the set of roles depending on the verb the token depends on. This may be particularly useful for distinguish different sentence settings and thus, whether a token denotes an event in a particular sentence type.

The system consists of two main processes. Firstly, given TimeML annotated text, it obtains the defined features plus the IOB2 tags of the annotated events. Then, using this data the system learns (trains) a model for event recognition and a model for event classification. Secondly, given plain text, it automatically gets the defined features using the described tools. With this data, the system applies the learned models to recognize and classify TimeML events.

4 Evaluation

In this section, firstly, the corpus, criteria and measures are defined. Secondly, the results obtained by our approach are presented. After that, the contribution of our approach is measured through different experiments: (1) general contribution, (2) semantic roles contribution, and (3) CRFs contribution. And finally, our approach is compared to the state of the art systems.

4.1 Corpus, criteria and measures

For the evaluation, the TimeBank 1.2 corpus (7881 events) was used without modification. All the results reported in this evaluation were obtained using a 5-fold cross validation. The n-fold train-test sets were built sorting the corpus files alphabetically and then sequentially select each set regarding the documents size. It is important to highlight the latter because if the n-folds were made regarding the number of documents, the sets had not been homogeneous due to the differences in TimeBank document sizes.

Only annotations matching the exact event span were considered as *correct* in recognition and classification, requiring also the class matching in the second case.

The following measures were used to score the evaluated approaches.

- Precision $\frac{correct\_annotations}{total\_approach\_annotations}$
- Recall $\frac{correct\_annotation}{total\_corpus\_annotations}$
- $F_{\beta=1}$ $\frac{2 * precision * recall}{precision + recall}$

4.2 Our approach results

Table 1 shows the results obtained by our approach for both recognition and classification of events. The last column (BF) indicates the best $F_{\beta=1}$ results obtained in the individual folds.

	Precision	**Recall**	**$F_{\beta=1}$**	BF
Recognition	83.43	79.54	**81.40**	82.43
Classification	68.84	60.15	**64.20**	69.68

Table 1: Our approach (CRF+Roles) results

The results show a high $F_{\beta=1}$ score in both recognition and classification, showing a good balance between precision and recall. This indicates that our approach is appropriate to address this task.

Focusing on classification task, Table 2 shows the detailed scores for each event class.

Looking at the specific class results, *reporting* obtained the best results. This is due to the fact that 80% of *reporting* events are represented by lemmas “say” and “report” with PoS “VBD” and “VBZ”. *Occurrence*, *perception*, *aspectual* and *i_state* obtained classification results over 50%.

Class (instances)	Precision	Recall	$F_{\beta=1}$
Reporting (1021)	91.90	89.18	90.51
Perception (48)	65.93	66.83	66.37
Aspectual (258)	81.35	47.00	59.57
I_Action (673)	51.40	29.30	37.32
I_State (582)	68.44	43.70	53.34
State (1107)	50.01	24.84	33.19
Occurrence (4192)	66.73	72.07	69.29

Table 2: CRF+Roles 5-fold detailed results

Although *perception* and *aspectual* are quite restricted to some lemmas, they obtained results below *reporting*. This is due to the fact that TimeBank contains very few examples of these classes. *I_action* and *state* show poorer results. In the case of the former, this is because some non-intensional verbs (e.g., "look") appear in the corpus as *i_action* under certain conditions, for example, when there is modality or these verbs appear in conditional sentences. This suggests the necessity of incorporating a word sense disambiguation (WSD) technique. Our approach did not take into account this information and thus the results are lower for this event class. In the case of *state*, the reasons for the low performance are the richness of this event class by means of lemmas, PoS, and phrases.

Finally, Table 3 shows the results of our approach by word class.

		Precision	Recall	$F_{\beta=1}$
Recognition	*Verb*	91.56	92.15	91.33
	Noun	72.67	48.26	58.42
	Adj.	66.78	38.09	48.35
Classification	*Verb*	73.86	74.21	73.51
	Noun	62.73	41.33	49.53
	Adj.	55.69	31.12	40.41

Table 3: CRF+Roles 5-fold word class results

It may be seen that the best results in both recognition and classification are obtained in verb events, followed by noun and adjective.

4.3 Contribution analysis

This subsection details the contribution of each aspect of our approach through three comparative experiments.

First experiment: general contribution

This experiment measures the general contribution of our approach by comparing its results with a baseline. TimeBank was analyzed to find a basic general rule to annotate events. The events are mainly denoted by verbs, pertaining to *occurrence* class. Hence, we propose a baseline that annotates all verbs as *occurrence* events. Table 4 shows results obtained by this baseline for both recognition and classification of events.

		Prec.	Recall	$F_{\beta=1}$
Our approach	*Recog.*	83.43	79.54	81.40
	Class.	68.84	60.15	64.20
Baseline	*Recog.*	72.50	65.20	68.60
	Class.	46.01	53.19	49.34

Table 4: Our approach vs Baseline results

Given the simplicity of the baseline, the results obtained are quite high. However, our approach $F_{\beta=1}$ significantly improves baseline by 19% for recognition and 30% for classification.

Second experiment: roles contribution

The main objective of this paper is to determine the impact of semantic roles in this task. To quantify it, a non-roles version of our approach was evaluated. This version only uses the general features described in section 3. Table 5 shows the results obtained.

		Precision	Recall	$F_{\beta=1}$
Our approach	*Recog.*	83.43	79.54	81.40
	Class.	68.84	60.15	64.20
Non-roles	*Recog.*	82.96	74.81	78.67
	Class.	67.53	54.80	60.50

Table 5: Our approach vs Non-roles results

Comparing these results with the ones obtained by our full featured approach, the application of roles improved especially the recall. Specifically, recall improved by 6% and 10% for recognition and classification respectively. The main improvement was achieved by *state* and *occurrence* classes (60% of the total improvement), especially, nominal events of that classes that concentrate around the 70% of the total contribution.

To illustrate corpus examples that have been improved by roles, Example 5 shows two sentences containing state events that were correctly tagged by the roles approach and missed by the

non-roles. In the examples, the TimeML events annotation and below the semantic roles annotation is reported.

(5) "There are still few buyers and the mood is <EVENT class=STATE>gloomy</EVENT>"
"[There A0] are [still AM-TMP] [few buyers A1] and [the mood A0] is [gloomy AM-MNR]"

"Security is now <EVENT>better</EVENT>"
"[Security A0] is [now AM-TMP] [better AM-MNR]"

In these cases, AM-MNR role information lead to a correct *state* event recognition.

Third experiment: CRFs contribution

In order to measure the CRFs contribution to this task, an extra experiment was carried out. This consisted of comparing, under the same setting, CRFs with a popular learning technique: support vector machines (SVM). As in Bethard and Martin (2006), YamCha[4] software was used (parameters: C=1 and polynomial_degree=2).

Table 6 shows the results obtained by the SVM-based approach in recognition and Table 7 reports the improvement (CRFs over SVM) distribution in the different word classes.

	Precision	**Recall**	**F$_{\beta=1}$**
Our approach (CRF)	83.43	79.54	81.40
SVM	80.00	75.10	77.40

Table 6: Our approach (CRF) vs SVM results

	Verb	**Noun**	**Adj.**	**Adv.**	**Prep.**
General	22%	71%	5%	1%	1%

Table 7: CRF improvement distribution among the word classes

These results verify that CRF improves SVM F$_{\beta=1}$ by 5% in this task. Furthermore, especially noun events take advantage of using CRF.

Finally, Figure 2 illustrates the results of our approach over the described experiments.

4.4 Comparison with the state of the art

Most systems found in the literature are data-driven approaches using morphosyntactic features. SVM based approaches (Bethard and Martin, 2006; March and Baldwin, 2008) achieved,

[4]http://chasen.org/~taku/software/YamCha/

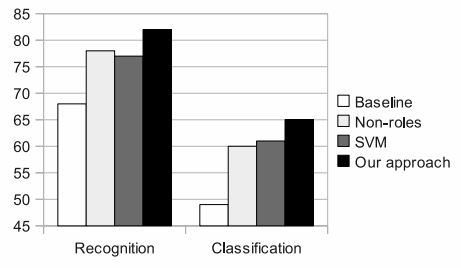


Figure 2: F$_{\beta=1}$ Results

approximately, 76% and 58% F$_{\beta=1}$ in event recognition and classification respectively. Boguraev and Ando (2005) used a robust risk minimization classifier to address this task and obtained 78.6% and 61% (without exploiting unannotated data). These results are very similar to the ones obtained by our non-roles approach. This suggests that using, apart from morphosyntactic features, additional features based on semantic roles could improve the approaches.

EVITA system (Saurí et al., 2005) combines linguistic and statistical techniques. On the one hand, it consists of a set of manually encoded rules based on morphosyntactic information. On the other hand, it includes a Bayesian learned disambiguation module to identify nominal events. The later was trained and tested using the whole corpus, therefore, the results could be inflated by this fact. For that reason, Bethard and Martin (2006) presented an EVITA implementation (Sim-Evita) to compare the results. Sim-Evita obtains an 73% and 51% F$_{\beta=1}$ in event recognition and classification respectively. These results suggest that data-driven improve rule-based approaches.

Only STEP evaluation showed detailed classification results. We agree that *state* events are the most complex and heterogeneous ones. Focusing on such events, our F$_{\beta=1}$ results (33%) improve Bethard's (25%) by 32%. Regarding the results obtained for each word class. Bethard's results presented good performance on classifying verb events (71%), but lower results in noun events (34%). Our approach results for noun events (49%) improve theirs by 44%. This suggests that the application of semantic roles enables our approach on making more general predictions. In this manner, our system may recog-

nize unseen nominal event instances as long as they share, with the seen instances, some semantic roles features.

5 Conclusions and Further Work

This paper presented an approach for the recognition and classification of TimeML events consisting of a CRF model learned using semantic roles as main feature. In addition to morphosyntactic features, the model was enhanced including extra semantic information, semantic role labeling, used for other applications with satisfactory results, but never employed before for this purpose. Our proposal was evaluated using the gold standard corpus, TimeBank 1.2, and the results obtained were analyzed and compared to measure the impact of both semantic roles and CRFs in the described task.

The obtained $F_{\beta=1}$ results demonstrated that semantic roles are useful to recognize (81.43%) and classify (64.20%) TimeML events, improving the presented baseline by 19% for recognition and 30% for classification. Specifically, Semantic roles employed as additional feature improved the recall of the non-roles version by 6% and 10% for recognition and classification respectively. This indicates that roles features led to more general models capable of better annotating unseen instances. The roles contribution was more significant in *state* and *occurrence* classes of noun events, concentrating around the 70% of the improvement.

Furthermore, it was verified that CRFs achieve higher results than models learned using other ML techniques such as SVM (5% improvement), contributing especially to nominal events. This demonstrated that CRF models are appropriate to face the task.

Finally, to the extent our results are comparable to state of the art evaluations, ours outperform the $F_{\beta=1}$ scores in both recognition and classification. Especially, our approach showed better performance than related works in *state* (32% improvement) and nominal events (44% improvement). Hence, the extension of the current approaches with semantic roles features could benefit their performance.

The main difficulties found in the task addressed in this paper are related to *i_action* and *state* events. In the former, we detected that modality and the word senses are important and must be treated to distinguish such events. In the later, although they were improved by our approach, *state* events are still the most complex class of events due to their richness in contrast to the reduced size of the training data. We agree with related literature that event classification results are still below other tasks performance, which indicates that this task is inherently complex and more training data may lead to significant improvements.

As further work we propose, firstly, improving the *i_action* results by taking into account the modality considering the *AM-MOD* role, and the word senses using a WSD technique. Secondly, the application of FrameNet role set (finer granularity) to determine which kind of roles are better to improve the current event annotation systems.

Acknowledgments

This paper has been supported by the Spanish Government, projects TIN-2006-15265-C06-01, TIN-2009-13391-C04-01 and PROMETEO/2009/119, where Hector Llorens is funded (BES-2007-16256).

References

Allan, James. 2002. *Topic Detection and Tracking: Event-Based Information Organization*. Kluwer Academic Publishers, Norwell, MA, USA.

Baker, Collin F., Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *COLING-ACL*, pages 86–90.

Bethard, Steven and James H. Martin. 2006. Identification of event mentions and their semantic class. In *EMNLP: Proceedings of the Conference on Empirical Methods in NLP*, pages 146–154. ACL.

Boguraev, Branimir and Rie Kubota Ando. 2005. Effective Use of TimeBank for TimeML Analysis. In *Annotating, Extracting and Reasoning about Time and Events 05151*.

Charniak, Eugene and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *43rd Annual Meeting of the ACL*.

Daniel, Naomi, Dragomir Radev, and Timothy Allison. 2003. Sub-event based multi-document summariza-

tion. In *HLT-NAACL Text summarization workshop*, pages 9–16. ACL.

Fellbaum, Christiane. 1998. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. MIT Press.

Gildea, Daniel and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Hagège, Caroline and Xavier Tannier. 2007. XRCE-T: XIP temporal module for TempEval campaign. In *TempEval (SemEval)*, pages 492–495, Prague, Czech Republic. ACL.

Lafferty, John D., Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th ICML*, pages 282–289. Morgan Kaufmann.

Llorens, Hector, Borja Navarro, and Estela Saquete. 2009. Using Semantic Networks to Identify Temporal Expressions from Semantic Roles. In *VI RANLP*, pages 219–224.

March, Olivia and Timothy Baldwin. 2008. Automatic event reference identification. In *ALTA 2008*, pages 79–87, Australia.

McCallum, Andrew and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *HLT-NAACL*, pages 188–191.

Melli, G., Y. Liu Z. Shi, Y. Wang, and F. Popowich. 2006. Description of SQUASH, the SFU Question Answering Summary Handler for the DUC-2006 Summarization Task. In *DUC*.

Palmer, Martha, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31.

Punyakanok, Vasin, Dan Roth, W. Yih, D. Zimak, and Y. Tu. 2004. Semantic role labeling via generalized inference over classifiers. In *HLT-NAACL (CoNLL)*, pages 130–133. ACL.

Pustejovsky, James, José M. Castaño, Robert Ingria, Roser Saurí, Robert Gaizauskas, Andrea Setzer, and Graham Katz. 2003a. TimeML: Robust Specification of Event and Temporal Expressions in Text. In *IWCS-5*.

Pustejovsky, James, Patrik Hanks, Roser Saurí, A. See, Robert Gaizauskas, Andrea Setzer, Dragomir R. Radev, Beth Sundheim, David Day, Lisa Ferro, and M. Lazo. 2003b. The TIMEBANK Corpus. In *Corpus Linguistics*, pages 647–656.

Pustejovsky, James. 2002. TERQAS: Time and Event Recognition for Question Answering Systems. In *ARDA Workshop*.

Saurí, Roser, Robert Knippen, Marc Verhagen, and James Pustejovsky. 2005. Evita: A robust event recognizer for qa systems. In *HLT/EMNLP*. ACL.

Schmid, Helmut. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, pages 44–49.

Setzer, Andrea and Robert Gaizauskas. 2000. Annotating Events and Temporal Information in Newswire Texts. In *LREC 2000*, pages 1287–1294.

Sha, Fei and Fernando C. N. Pereira. 2003. Shallow parsing with conditional random fields. In *HLT-NAACL*.

Tenny, Carol and James Pustejovsky. 2000. *Events as Grammatical Objects. The Converging Perspectives of Lexical Semantics and Syntax*. CSLI.

Vendler, Zeno, 1967. *Linguistics and philosophy*, chapter Verbs and times, pages 97–121. Cornell University Press, Ithaca, NY.

Exploiting Structured Ontology to Organize Scattered Online Opinions

Yue Lu, Huizhong Duan, Hongning Wang, ChengXiang Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign
{yuelu2,duan9,wang296,czhai}@illinois.edu

Abstract

We study the problem of integrating scattered online opinions. For this purpose, we propose to exploit structured ontology to obtain well-formed relevant aspects to a topic and use them to organize scattered opinions to generate a structured summary. Particularly, we focus on two main challenges in implementing this idea, (1) how to select the most useful aspects from a large number of aspects in the ontology and (2) how to order the selected aspects to optimize the readability of the structured summary. We propose and explore several methods for solving these challenges. Experimental results on two different data sets (US Presidents and Digital Cameras) show that the proposed methods are effective for selecting aspects that can represent the major opinions and for generating coherent ordering of aspects.

1 Introduction

The explosive growth of online opinions raises interesting challenges for opinion integration and summarization. It is especially interesting to integrate and summarize scattered opinions in blog articles and forums as they tend to represent the general opinions of a large number of people and get refreshed quickly as people dynamically generate new content, making them valuable for understanding the current views of a topic.

However, opinions in blogs and forums are usually fragmental, scattered around, and buried among other off-topic content, so it is quite challenging to organize them in a meaningful way. Traditional text summarization techniques generate an unstructured list of sentences as a summary, which cannot reveal representative opinions on different aspects of a topic or effectively facilitate navigation into the huge opinion space. To address this limitation, recent work has shown the usefulness of generating a structured summary of opinions, in which related opinions are grouped into topical aspects with explicit labeling of all the aspects. A major challenge in producing such a structured summary is how to generate these aspects for an arbitrary topic (e.g., products, political figures, policies, etc.). Intuitively, the aspects should be concise phrases that can both be easily interpreted in the context of the topic under consideration and capture the major opinions. However, where can we find such phrases and which phrases should we select as aspects? Furthermore, once we selected aspects, how should we order them to improve the readability of a structured summary? One way to generate aspects is to cluster all the opinion sentences and then identify representative phrases in each cluster. Although aspects selected in this way can effectively capture the major opinions, a major limitation is that it is generally hard to ensure that the selected phrases are well connected with the given topic (Chen and Dumais, 2000).

In this paper, we propose a novel approach to generating aspects by leveraging the ontologies with structured information that are available online, such as open domain knowledge base in Freebase[1]. Such kind of ontology data is not in small scale by any measure. For example, Freebase alone contains more than 10 million topics, 3000 types, and 30,000 properties; moreover, it is constantly growing as people collaboratively contribute. Freebase provides different properties for different types of topics such as personal information for a "US President" and product features for a "Digital Camera". Since this kind of resources can provide related entities/relations for a

[1] http://www.freebase.com

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 734–742,
Beijing, August 2010

wide range of topics , our general idea is to leverage them as guidance for more informed organization of scattered online opinions, and in particular, to select the most important properties of a topic from such structured ontology as aspects to generate a structured opinion summary. A significant advantage of this approach to aspect generation is that the selected aspects are guaranteed to be very well connected with the topic, but it also raises an additional challenge in selecting the aspects to best capture the major opinions from a large number of aspects provided for each topic in the ontology. Different from some existing work on exploiting ontologies, e.g., (Sauper and Barzilay, 2009), which relies on training data, we focus on exploring *unsupervised* approaches, which can be applied to a larger scope of topics.

Specifically, given a topic with entries in an ontology and a collection of scattered online opinions about the topic, our goal is to generate a structured summary where representative major opinions are organized with well aligned aspects and in an order easy for human to follow. We propose the following general approach: First, retrieval techniques are employed to align opinions to relevant aspects. Second, a subset of most interesting aspects are selected. Third, we will further order the selected aspects to present them in a reasonable order. Finally, for the opinions uncovered by the selected aspects from the ontology, we use a phrase ranking method to suggest new aspects to add to the ontology for increasing its coverage.

Implementing the second and third steps involves special challenges. In particular, without any training data, it is unclear how we should show the most interesting aspects in ontology with major opinions aligned and which presentation order of aspects is natural and intuitive for human. Solving these two challenges is the main focus of this paper. We propose three methods for aspect selection, i.e., size-based, opinion coverage-based, and conditional entropy-based methods, and two methods for aspect ordering, i.e., ontology-ordering and coherence ordering. We evaluate our methods on two different types of topics: US Presidents and Digital Cameras. Qualitative results demonstrate the utility of integrating opinions based on structured ontology as well as the generalizability of proposed methods. Quantitative evaluation is also conducted to show the effectiveness of our methods.

Note that we use the term "opinion" to broadly refer to any discussion in opinionated sources such as blogs and reviews. This allows us to formulate and solve the problem in a general way. Indeed, the main goal of our work is to extract and organize the major opinions about a topic that are buried in many scattered opinionated sources rather than perform deeper understanding of opinions (e.g., distinguishing positive from negative opinions), which can be done by using any existing sentiment analysis technique as an orthogonal post-processing step after applying our method.

2 Related Work

Aspect summarization, i.e., structured opinion summarization over topical aspects, has attracted much attention recently. Existing work identifies aspects using frequent-pattern/association-rule mining, e.g. (Liu et al., 2005; Popescu and Etzioni, 2005), sentence clustering, e.g. (Gamon et al., 2005; Leouski and Croft, 1996), or topic modeling, e.g. (Mei et al., 2006; Titov and McDonald, 2008). After that, meaningful and prominent phrases need to be selected to represent the aspects, e.g. (Zhao and He, 2006; Mei et al., 2007). However, these methods suffer from the problem of producing trivial aspects. Consequently, some of the aspects generated are very difficult to interpret (Chen and Dumais, 2000). In this paper, we propose a different kind of approach that is to use aspects provided by ontology which are known to be relevant and easy to interpret.

Ontology is used in (Carenini et al., 2005) but only for mapping product features. The closest work to ours are (Lu and Zhai, 2008; Sauper and Barzilay, 2009); both try to use well-written articles for summarization. However, (Lu and Zhai, 2008) assumes the well-written article is structured with explicit or implicit aspect information, which does not always hold in practice, while (Sauper and Barzilay, 2009) needs a relatively large amount of training data in the given domain. In comparison, our work only needs the ontology information for the given topic which is much easier to obtain from resources such as Freebase.

3 Methods

Given (1) an input topic T, (2) a large number of aspects/properties $A = \{A_1, ..., A_m\}$ from an ontology that are related to T, and (3) a huge collection of scattered opinion sentences about the topic $D_T = \{s_1, \ldots, s_n\}$, our goal is to generate a structured organization of opinions that are both aligned well with the interesting aspects and representative of major opinions about the topic.

The envisioned structured organization consists of a sequence of selected aspects from ontology ordered to optimize readability and a set of sentences matching each selected aspect. Once we obtain a set of sentences in each aspect, we can easily apply a standard text summarization method to further summarize these sentences, thus the unique challenges related to our main idea of exploiting ontology are the following, which are also the main focus of our study:

Aspect Selection: How can we select a subset of aspects $A' \subset A$ to capture the *major* opinions in our opinion set D_T?

Aspect Ordering: How can we order a subset of selected aspects A' so as to present them in an order $\pi(A')$ that is most natural with respect to human perception?

New Aspects Suggestion: Can we exploit the opinions in D_T to suggest new aspects to be added to the ontology?

3.1 Aspect Selection

In order to align the scattered opinions to the most relevant aspects, we first use each aspect label $A_i \in A$ as a query to retrieve a set of relevant opinions in the collection $S_i \subseteq D_T$ with a standard language modeling approach, i.e., the KL-divergence retrieval model (Zhai and Lafferty, 2001). Up to 1000 opinion sentences are retrieved for each aspect; each opinion sentence can be potentially aligned to several aspects. In this way, scattered online discussion are linked to the most relevant aspects in the ontology, which enables a user to use aspects as "semantic bridges" to navigate into the opinion space..

However, there are usually a lot of candidate aspects in an ontology, and only some are heavily commented in online discussions, so showing all the aspects is not only unnecessary, but also overwhelming for users. To solve this problem, we propose to utilize the aligned opinions to further select a subset of the most interesting aspects $A' \subset A$ with size k. Several approaches are possible for this subset selection problem.

Size-based: Intuitively, the selected subset A' should reflect the major opinions. So a straightforward method is to order the aspects A_i by the size of the aligned opinion sentences S_i, i.e., the number of relevant opinion sentences, and then select the top k ones.

Opinion Coverage-based: The previous method does not consider possible redundancy among the aspects. A better approach is to select the subset that covers as many *distinct* opinion sentences as possible. This can be formulated as a maximum coverage problem, for which a greedy algorithm is known to be a good approximation: we select one aspect at a time that is aligned with the largest number of uncovered sentences.

Conditional Entropy-based: Aspects from a structured ontology are generally quite meaningful, but they are not designed specifically for organizing the opinions in our data set. Thus, they do not necessarily correspond well to the natural clusters in scattered opinions. To obtain aspects that are aligned well with the natural clusters in scattered opinions, we can first cluster D_T into l clusters $C = \{C_1, \ldots, C_l\}$ using K-means with $TF \times IDF$ as features, and then choose the subset of aspects that minimize Conditional Entropy of the cluster label given the aspect:

$$A' = \arg\min H(C|A') = \arg\min \left[- \sum_{A_i \in A', C_i \in C} p(A_i, C_i) \log \frac{p(A_i, C_i)}{p(A_i)} \right]$$

This Conditional Entropy measures the uncertainty about the cluster label of a sentence given the knowledge of its aspect. Intuitively, if the aspects are aligned well with the clusters, we would be able to predict well the cluster label of a sentence if we know its aspect, thus there would be less uncertainty about the cluster label. In the extreme case when the cluster label can be completely determined by the aspect, the conditional entropy would reach its minimum (i.e., 0). Intuitively, the conditional entropy-based method essentially selects the most appropriate aspects from

Algorithm 1 Greedy Algorithm for Conditional Entropy Based Aspect Selection

Input: $A = \{A_1, ..., A_m\}$
Output: k-sized $A' \subseteq A$

$A' = \{\cup_{i=1}^{m} A_i\}$
for j=1 to k **do**
  $bestH = \infty$; $bestA = A_0$
  for each A_i in A **do**
    $tempA' = \{A_i, A' \setminus A_i\}$
    if $H(C|tempA') < bestH$ **then**
      $bestH = H(C|tempA')$
      $bestA = A_i$
  $A' = \{bestA, A' \setminus bestA\}$
output A'

the ontology to label clusters of opinions.

The exact solution of this combinatorial optimization problem is NP-complete, so we employ a polynomial time greedy algorithm to approximate it: in the i-th iteration, we select the aspect that can minimize the conditional entropy given the previous $i-1$ selected aspects. Pseudo code is given in Algorithm 1.

3.2 Aspect Ordering

In order to present the selected aspects to users in a most natural way, it is important to obtain a coherent order of them, i.e., generating an order consistent with human perception. To achieve this goal, our idea is to use human written articles on the topic to learn how to organize the aspects automatically. Specifically, we would order aspects so that the relative order of the sentences in all the aspects would be as consistent with their order in the original online discussions as possible.

Formally, the input is a subset of selected aspects A'; each $A_i \in A'$ is aligned with a set of relevant opinion sentences $S_i = \{S_{i,1}, S_{i,2}, ...\}$. We define a coherence measurement function over sentence pairs $Co(S_{i,k}, S_{j,l})$, which is set to 1 iff $S_{i,k}$ appears before $S_{j,l}$ in the same article. Otherwise, it is set to 0. Then a coherence measurement function over an aspect pair can be calculated as

$$Co(A_i, A_j) = \frac{\sum_{S_{i,k} \in S_i, S_{j,l} \in S_j} Co(S_{i,k}, S_{j,l})}{|S_i||S_j|}$$

As an output, we would like to find a permutation $\hat{\pi}(A')$ that maximizes the coherence of all pairwise aspects, i.e.,

$$\hat{\pi}(A') = \arg\max_{\pi(A')} \sum_{A_i, A_j \in A', A_i \prec A_j} Co(A_i, A_j)$$

Algorithm 2 Greedy Algorithm for Coherence Based Aspect Ordering

Input: A
Output: $\pi(A)$

for each A_i, A_j in A **do**
  calculate $Co(A_i, A_j)$
for $p = 1$ to $len = A.size()$ **do**
  $Max = A[1]$
  for each aspect A_i in A **do**
    $A_i.coherence = 0$
    for each aspect A_j in $\pi(A)$ **do**
      $A_i.coherence += Co(A_j, A_i)$
    for each aspect A_j in A, $j \neq i$ **do**
      $A_i.coherence += Co(A_i, A_j)$
    if $A_i.coherence > Max.coherence$ **then**
      $Max = A_i$
  remove Max from A; add Max to $\pi(A)$
output $\pi(A)$

where $A_i \prec A_j$ means that A_i is before A_j. It is easy to prove that the problem is NP-complete. Therefore, we resort to greedy algorithms to find approximations of the solution. Particularly we view the problem as a ranking problem. The algorithm proceeds by finding at each ranking position an aspect that can maximize the coherence measurement, starting from the top of the rank list. The detailed algorithm is given in Algorithm 2.

3.3 New Aspects Suggestion

Finally, if the opinions cover more aspects than in the ontology, we also want to identify informative phrases to label such extra aspects; such phrases can also be used to further augment the ontology with new aspects.

This problem is similar to existing work on generating labels for clusters (Zeng et al., 2004) or topic models (Mei et al., 2007). Here we employ a simple but representative technique to demonstrate the feasibility of discovering interesting new aspects for augmenting the ontology. We first extract named entities from scattered opinions D_T using Stanford Named Entity Recognizer (Finkel et al., 2005). After that, we rank the phrases by pointwise Mutual Information (MI):

$$MI(T, ph) = \log \frac{P(T, ph)}{P(T)P(ph)}$$

where T is the given topic and ph refers to a candidate entity phrase. $P(T, ph)$ is proportional to the number of opinion sentences they co-occur; $P(T)$ or $P(ph)$ are proportional to the number of times T or ph appears. A higher MI value indicates a

Statistics	Category 1 US president	Category 2 Digital Camera
Number of Topics	36	110
Number of Aspects	65±26	32±4
Number of Opinions	1001±1542	170±249

Table 1: Statistics of Data Sets

stronger association. We can then suggest the top ranked entity phrases that are not in the selected aspects as new aspects.

4 Experiments

4.1 Data Sets

To examine the generalizability of our methods, we test on two very different categories of topics: *US Presidents* and *Digital Cameras*.[2] For the ontology, we leverage Freebase, downloading the structured ontology for each topic. For the opinion corpus, we use blog data for US Presidents and customer reviews for Digital Cameras. The blog entries for US Presidents were collected by using Google Blog Search[3] with the name of a president as the query. Customer reviews for Digital Cameras were crawled from CNET[4]. The basic statistics of our data sets is shown in Table 1. For all the data collections, Porter stemmer (Porter, 1997) is applied and stop words are removed.

4.2 Sample Results

We first show sample results of automatic organization of online opinions. We use the opinion coverage-based algorithm to select 10 aspects (10-20 aspects were found to be optimal in (Käki, 2005)) and then apply the coherence-based aspect ordering method. The number of clusters is set so that there are on average 15 opinions per cluster.

Opinion Organization: Table 2 and Table 3 present sample results for President Ronald Reagan and Sony Cybershot DSC-W200 camera respectively[5]. We can see that (1) although Freebase aspects provide objective and accurate information about the given topics, extracted opinion sentences offer additional subjective information; (2) aligning scattered opinion sentences to most relevant aspects in the ontology helps digestion and navigation; and (3) the support number, which is the number of opinion sentences aligned to an aspect, can show the popularity of the aspect in the online discussions.

Adaptability of Aspect Selection: Being unsupervised is a significant advantage of our methods over most existing work. It provides flexibility of applying the methods in different domains without the requirement of training data, benefiting from both the ontology based template guidance as well as data-driven approaches. As a result, we can generate different results for different topics even in the same domain. In Table 4, we show the top three selected and ordered aspects for Abraham Lincoln and Richard Nixon. Although they belong to the same category, different aspects are picked up due to the differences in online opinions. People talk a lot about Lincoln's role in American Civil War and his famous quotation, but when talking about Nixon, people focus on ending the Vietnam war and the Watergate scandal. "Date of birth" and "Government position" are ranked first because people tend to start talking from these aspects, which is more natural than starting from aspects like "Place of death".

Baseline Comparison: We also show below the aspects for Lincoln generated by a representative approach using clustering method (e.g. (Gamon et al., 2005)). i.e., we label the largest clusters by selecting phrases with top mutual information. We can see that although some phrases make sense, not all are well connected with the given topic; using aspects in ontology circumvents this problem. This example confirms the finding in previous work that the popular existing clustering-based approach to aspects generation cannot generate meaningful labels (Chen and Dumais, 2000).

```
Vincent
New Salem State Historic Site
USS Abraham Lincoln
Martin Luther King Jr
Gettysburg
John F.
```

New Aspect Discovery: Finally, in Table 5 we show some phrases ranked among top 10 using the method described in Section 3.3. They reveal additional aspects covered in online discussions and serve as candidate new aspects to be added to Freebase. Interestingly, John Wilkes Booth, who assassinated President Lincoln, is not explicitly

[2]We have made our data sets available at `http://timan.cs.uiuc.edu/downloads.html`.

[3]`http://blogsearch.google.com`

[4]`http://www.cnet.com`

[5]Due to space limit, we only show the first few aspects as output by our methods.

FreeBase Aspects	Supt	Representative Opinion Sentences
Appointees: - Martin Feldstein - Chief Economic Advisor	897	Martin Feldstein, whose criticism of Reagan era deficits has not been forgotten. Reagan's first National Security advisor was quoted as declaring...
Government Positions Held: - President of the United States - Jan 20, 1981 to Jan 20, 1989	967	1981 Jan 20, Ronald Reagan was sworn in as president as 52 American hostages boarded a plane in Tehran and headed toward freedom. 40th president of the US Ronald Reagan broke the so called "20 year curse"...
Vice president: - George H. W. Bush	847	8 years, 1981-1988 George H. W. Bush as vice president under Ronald Reagan... ...exception to the rule was in 1976, when George H W Bush beat Ronald.

Table 2: Opinion Organization Result for President Ronald Reagan

FreeBase Aspects	Supt	Representative Opinion Sentences
Format: - Compact	13	Quality pictures in a compact package. ... amazing is that this is such a small and compact unit but packs so much power.
Supported Storage Types: - Memory Stick Duo	11	This camera can use Memory Stick Pro Duo up to 8 GB Using a universal storage card and cable (c'mon Sony)
Sensor type: - CCD	10	I think the larger ccd makes a difference. but remember this is a small CCD in a compact point-and-shoot.
Digital zoom: -2×	47	once the digital :smart" zoom kicks in you get another 3x of zoom I would like a higher optical zoom, the W200 does a great digital zoom translation...

Table 3: Opinion Organization Result for Sony Cybershot DSC-W200 Camera

listed in Freebase, but we can find it in people's online discussion using mutual information.

4.3 Evaluation of Aspect Selection

Measures: Aspect selection is a new challenge, so there is no standard way to evaluate it. It is also very hard for human to read all of the aspects and opinions and then select a gold standard subset. Therefore, we opt to use indirect measures capturing different characteristics of the aspect selection problem (1) *Aspect Coverage* (AC): we first assign each aspect A_i to the cluster C_j that has the most overlapping sentences with A_i, approximating the cluster that would come into mind when a reader sees A_i. Then AC is defined as the percentage of the clusters covered by at least one aspect. (2) *Aspect Precision* (AP): for each covered cluster C_i, AP measures the Jaccard similarity between C_i as a set of opinions and the union of all aspects assigned to C_i. (3) *Average Aspect Precision* (AAP): defines averaged AP for all clusters where an uncovered C_i has a zero AP; it essentially combines AC and AP. We also report *Sentence Coverage* (SC), i.e., how many distinct opinion sentences can be covered by the selected aspects and *Conditional Entropy* (H), i.e., how well the selected aspects align with the natural clusters in the opinions; a smaller H value indicates a better alignment.

Results: We summarize the evaluation results in Table 6. In addition to the three methods described in Section 3.1, we also include one baseline of averaging 10 runs of random selection. The best performance by each measure on each data set is highlighted in bold font. Not surprisingly, opinion coverage-based approach has the best sentence coverage (SC) performance and conditional entropy-based greedy algorithm achieves the lowest H. Size-based approach is best in aspect precision but at the cost of lowest aspect coverage. The trade-off between AP and AC is comparable to that between precision and recall as in information retrieval while AAP summarizes the combination of these two. The greedy algorithm based on conditional entropy outperforms all other approaches in AC and also in AAP, suggesting that it can provide a good balance between AP and AC.

Measures	SC	H	AC	AP	AAP
PRESIDENTS					
Random	503	1.9069	0.5140	0.0933	0.1223
Size-based	500	1.9656	0.3108	**0.1508**	0.0949
Opin Cover	**746**	1.8852	0.5463	0.0913	0.1316
Cond Ent.	479	**1.7687**	**0.5770**	0.0856	**0.1552**
CAMERAS					
Random	55	1.6389	0.6554	0.0871	0.1271
Size-based	70	1.6463	0.6071	**0.1077**	0.1340
Opin Cover	**82**	1.5866	0.6998	0.0914	0.1564
Cond Ent.	70	**1.5598**	**0.7497**	0.0789	**0.1574**

Table 6: Evaluation Results for Aspect Selection

Supt	**Richard-Nixon**	Supt	**Abraham-Lincoln**
50	Date of birth: - Jan 9, 1913	419	Government Positions Held: - United States Representative Mar 4,1847-Mar 3,1849
108	Tracks Recorded: - 23-73 Broadcast: End of the Vietnam War	558	Military Commands: - American Civil War - United States of America
120	Works Written About This Topic: - Watergate	810	Quotations: - Nearly all men can stand adversity, but if you want to test a man's character, give him power.

Table 4: Comparison of Aspect Selection for Two Presidents (aligned opinions are omitted here)

Suggested Phrases	**Supporting Opinion Sentences**
Abraham Lincoln Presidential Library	CDB projects include the Abraham Lincoln Presidential Library and Museum
Abraham Lincoln Memorial	..., eventually arriving at Abraham Lincoln Memorial.
John Wilkes Booth	John Wilkes Booth shoots President Abraham Lincoln at Ford's Theatre ...

Table 5: New Phrases for Abraham Lincoln

4.4 Evaluation of Aspect Ordering

Human Annotation: In order to quantitatively evaluate the effectiveness of aspect ordering, we conduct user studies to establish gold standard ordering. Three users were each given k selected aspects and asked to perform two tasks for each US President: (1) identify clusters of aspects that are more natural to be presented together (*cluster constraints*) and (2) identify aspect pairs where one aspect is preferred to appear before the other from the viewpoint of readability. (*order constraints*). We did not ask them to provide a full order of the k aspects, because we suspect that there are usually more than one "perfect" order. Instead, identifying partial orders or constraints is easier for human to perform, thus provides more robust gold standard.

Human Agreement: After obtaining the human annotation results, we first study human consensus on the ordering task. For both types of human identified constraints, we convert them into pair-wise relations of aspects, e.g., "A_i and A_j should be presented together" or "A_i should be displayed before A_j". Then we calculate the agreement percentage among the three users. In Table 7, we can see that only a very small percentage of pair-wise partial orders (15.92% of the cluster constraints and none of the order constraints) are agreed by all the three users, though the agreement of clustering is much higher than that of ordering. This indicates that ordering the aspects is a subjective and difficult task.

Measures: Given the human generated gold standard of partial constraints, we use the following measures to evaluate the automatically generated full ordering of aspects: (1) *Cluster Precision* (pr_c): for all the aspect pairs placed in the same cluster by human, we calculate the percentage of them that are also placed together in the system output. (2) *Cluster Penalty* (p_c): for each aspect pair placed in the same cluster by human, we give a linear penalty proportional to the number of aspects in between the pair that the system places; p_c can be interpreted as the average number of aspects between aspect pairs that should be presented together in the case of misordering. Smaller penalty corresponds to better ordering performance. (3) *Order Precision* (pr_o): the percentage of correctly predicted aspect pairs compared with human specified order.

AgreedBy	**Cluster Constraint**	**Order Constraint**
1	37.14%	89.22%
2	46.95%	10.78%
3	15.92%	0.00%

Table 7: Human Agreement on Ordering

Results: In Table 8, we report the ordering performance based on two selection algorithms: opinion coverage-based and conditional entropy-based. Different selection algorithms provide different subsets of aspects for the ordering algorithms to operate on. For comparison with our coherence-based ordering algorithm, we include a random baseline and Freebase ontology ordering. Note that Freebase order is a very strong baseline because it is edited by human even though the purpose was not for organizing opinions. To take into account the variation of human annotation, we use four versions of gold standard: three are from the individual annotators and one from the union of their annotation. We did not include the gold stan-

Selection Algo	Gold STD	Cluster Random	Precision Freebase	(pr_c) Coherence	Cluster Random	Penalty Freebase	(p_c) Coherence	Order Random	Precision Freebase	(pr_o) Coherence
Opin Cover	1	0.3290	**0.9547**	0.9505	1.8798	0.1547	**0.1068**	0.4804	**0.7059**	0.4510
Opin Cover	2	0.3266	**0.9293**	0.8838	1.7944	0.3283	**0.1818**	**0.4600**	0.4000	0.4000
Opin Cover	3	0.2038	**0.4550**	0.4417	2.5208	**1.3628**	1.7994	0.5202	0.4561	**0.5263**
Opin Cover	union	0.3234	**0.7859**	0.7237	1.8378	0.6346	**0.4609**	**0.4678**	0.4635	0.4526
Cond Entropy	1	0.2540	**0.9355**	0.8978	2.0656	0.2957	**0.2016**	0.5106	**0.7111**	0.5444
Cond Entropy	2	0.2535	0.7758	**0.8323**	2.1790	0.7530	**0.5222**	0.4759	**0.6759**	0.5093
Cond Entropy	3	0.2523	0.4030	**0.5545**	2.3079	2.1328	**1.1611**	0.5294	0.7143	**0.8175**
Cond Entropy	union	0.3067	0.7268	**0.7488**	1.9735	1.0720	**0.7196**	0.5006	0.6500	**0.6833**

Table 8: Evaluation Results on Aspect Ordering

dard that is the intersection of three annotators because that would leave us with too little overlap.

We have several observations: (1) In general, results show large variations when using different versions of gold standard, indicating the subjective nature of the ordering task. (2) Coherence-based ordering shows similar performance to Freebase order-based in cluster precision (pr_c), but when we take into consideration the distance-based penalty (p_c) of separating aspects pairs in the same cluster, coherence-based ordering is almost always significantly better except in one case. This shows that our method can effectively learn the coherence of aspects based on how their aligned opinion sentences are presented in online discussions. (3) Order precision (pr_o) can hardly distinguish different ordering algorithm. This indicates that people vary a lot in their preferences as which aspects should be presented first. However, in cases when the random baseline outperforms others the margin is fairly small, while Freebase order and coherence-based order have a much larger margin of improvement when showing superior performance.

5 Conclusions and Future Work

A major challenge in automatic integration of scattered online opinions is how to organize all the diverse opinions in a meaningful way for any given topic. In this paper, we propose to solve this challenge by exploiting related aspects in structured ontology which are guaranteed to be meaningful and well connected to the topic. We proposed three different methods for selecting a subset of aspects from the ontology that can best capture the major opinions, including size-based, opinion coverage-based, and conditional entropy-based methods. We also explored two ways to order aspects, i.e., ontology-order and coherence optimization. In addition, we also proposed appropriate measures for quantitative evaluation of both aspect selection and ordering.

Experimental evaluation on two data sets (US President and Digital Cameras) shows that by exploiting structured ontology, we can generate interesting aspects to organize scattered opinions. The conditional entropy method is shown to be most effective for aspect selection, and the coherence optimization method is more effective than ontology-order in optimizing the coherence of the aspect ordering, though ontology-order also appears to perform reasonably well. In addition, by extracting salient phrases from the major opinions that cannot be covered well by any aspect in an existing ontology, we can also discover interesting new aspects to extend the existing ontology.

Complementary with most existing summarization work, this work proposes a new direction of using structured information to organize and summarize unstructured opinions, opening up many interesting future research directions. For instance, in order to focus on studying aspect selection and ordering, we have not tried to optimize sentences matching with aspects in the ontology; it would be very interesting to further study how to accurately retrieve sentences matching each aspect. Another promising future work is to organize opinions using both structured ontology information and well-written overview articles.

Acknowledgment

We thank the anonymous reviewers for their useful comments. This paper is based upon work supported in part by an IBM Faculty Award, an Alfred P. Sloan Research Fellowship, an AFOSR MURI Grant FA9550-08-1-0265, and by the National Science Foundation under grants IIS-0347933, IIS-0713581, IIS-0713571, and CNS-0834709.

References

Carenini, Giuseppe, Raymond T. Ng, and Ed Zwart. 2005. Extracting knowledge from evaluative text. In *K-CAP '05: Proceedings of the 3rd international conference on Knowledge capture*, pages 11–18, New York, NY, USA. ACM.

Chen, Hao and Susan Dumais. 2000. Bringing order to the web: automatically categorizing search results. In *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 145–152, New York, NY, USA. ACM.

Finkel, Jenny Rose, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370, Morristown, NJ, USA. Association for Computational Linguistics.

Gamon, Michael, Anthony Aue, Simon Corston-Oliver, and Eric K. Ringger. 2005. Pulse: Mining customer opinions from free text. In Famili, A. Fazel, Joost N. Kok, José María Peña, Arno Siebes, and A. J. Feelders, editors, *IDA*, volume 3646 of *Lecture Notes in Computer Science*, pages 121–132. Springer.

Käki, Mika. 2005. Optimizing the number of search result categories. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 1517–1520, New York, NY, USA. ACM.

Leouski, Anton V. and W. Bruce Croft. 1996. An evaluation of techniques for clustering search results. Technical report.

Liu, Bing, Minqing Hu, and Junsheng Cheng. 2005. Opinion observer: analyzing and comparing opinions on the web. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 342–351, New York, NY, USA. ACM.

Lu, Yue and Chengxiang Zhai. 2008. Opinion integration through semi-supervised topic modeling. In Huai, Jinpeng, Robin Chen, Hsiao-Wuen Hon, Yunhao Liu, Wei-Ying Ma, Andrew Tomkins, and Xiaodong Zhang, editors, *WWW*, pages 121–130. ACM.

Mei, Qiaozhu, Chao Liu, Hang Su, and ChengXiang Zhai. 2006. A probabilistic approach to spatiotemporal theme pattern mining on weblogs. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 533–542.

Mei, Qiaozhu, Xuehua Shen, and ChengXiang Zhai. 2007. Automatic labeling of multinomial topic models. In Berkhin, Pavel, Rich Caruana, and Xindong Wu, editors, *KDD*, pages 490–499. ACM.

Pang, Bo and Lillian Lee. 2007. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

Popescu, Ana-Maria and Oren Etzioni. 2005. Extracting product features and opinions from reviews. In *HLT '05*, pages 339–346, Morristown, NJ, USA. Association for Computational Linguistics.

Porter, M. F. 1997. An algorithm for suffix stripping. pages 313–316.

Sauper, Christina and Regina Barzilay. 2009. Automatically generating wikipedia articles: A structure-aware approach. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 208–216, Suntec, Singapore, August. Association for Computational Linguistics.

Titov, Ivan and Ryan McDonald. 2008. Modeling online reviews with multi-grain topic models. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 111–120, New York, NY, USA. ACM.

Zeng, Hua-Jun, Qi-Cai He, Zheng Chen, Wei-Ying Ma, and Jinwen Ma. 2004. Learning to cluster web search results. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 210–217, New York, NY, USA. ACM.

Zhai, Chengxiang and John Lafferty. 2001. Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of CIKM 2001*, pages 403–410.

Zhao, Jing and Jing He. 2006. Learning to generate labels for organizing search results from a domain-specified corpus. In *WI '06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 390–396, Washington, DC, USA. IEEE Computer Society.

Enhancing Morphological Alignment for Translating Highly Inflected Languages *

Minh-Thang Luong
School of Computing
National University of Singapore
luongmin@comp.nus.edu.sg

Min-Yen Kan
School of Computing
National University of Singapore
kanmy@comp.nus.edu.sg

Abstract

We propose an unsupervised approach utilizing only raw corpora to enhance morphological alignment involving highly inflected languages. Our method focuses on *closed-class morphemes*, modeling their influence on nearby words. Our language-independent model recovers important links missing in the IBM Model 4 alignment and demonstrates improved end-to-end translations for English-Finnish and English-Hungarian.

1 Introduction

Modern statistical machine translation (SMT) systems, regardless of whether they are word-, phrase- or syntax-based, typically use the word as the atomic unit of translation. While this approach works when translating between languages with limited morphology such as English and French, it has been found inadequate for morphologically-rich languages like Arabic, Czech and Finnish (Lee, 2004; Goldwater and McClosky, 2005; Yang and Kirchhoff, 2006). As a result, a line of SMT research has worked to incorporate morphological analysis to gain access to information encoded within individual words.

In a typical MT process, word aligned data is fed as training data to create a translation model. In cases where a highly inflected language is involved, the current word-based alignment approaches produce low-quality alignment, as the statistical correspondences between source and target words are diffused over many morphological forms. This problem has a direct impact on end translation quality.

Our work addresses this shortcoming by proposing a morphologically sensitive approach to word alignment for language pairs involving a highly inflected language. In particular, our method focuses on a set of *closed-class morphemes* (CCMs), modeling their influence on nearby words. With the model, we correct erroneous alignments in the initial IBM Model 4 runs and add new alignments, which results in improved translation quality.

After reviewing related work, we give a case study for morpheme alignment in Section 3. Section 4 presents our four-step approach to construct and incorporate our CCM alignment model into the grow-diag process. Section 5 describes experiments, while Section 6 analyzes the system merits. We conclude with suggestions for future work.

2 Related Work

MT alignment has been an active research area. One can categorize previous approaches into those that use language-specific syntactic information and those that do not. Syntactic parse trees have been used to enhance alignment (Zhang and Gildea, 2005; Cherry and Lin, 2007; DeNero and Klein, 2007; Zhang et al., 2008; Haghighi et al., 2009). With syntactic knowledge, modeling long distance reordering is possible as the search space is confined to plausible syntactic variants. However, they generally require language-specific tools and annotated data, making such approaches infeasible for many languages. Works that follow non-syntactic approaches, such as (Matusov et al.,

*This work was supported by a National Research Foundation grant "Interactive Media Search" (grant # R-252-000-325-279)

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 743–751,
Beijing, August 2010

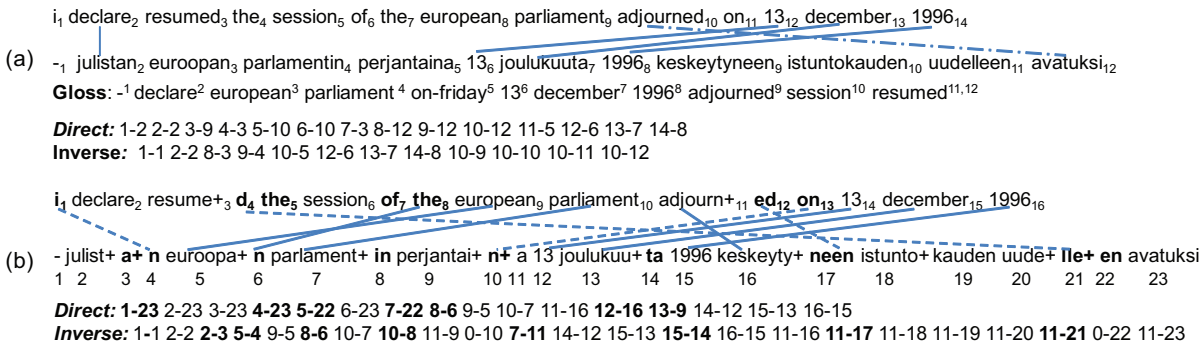


Figure 1: **Sample English-Finnish IBM Model 4 alignments**: (a) word-level and (b) morpheme-level. Solid lines indicate intersection alignments, while the exhaustive asymmetric alignments are listed below. In (a), translation glosses for Finnish are given; the dash-dot line is the incorrect alignment. In (b), bolded texts are closed-class morphemes (CCM), while bolded indices indicate alignments involving CCMs. The dotted lines are correct CCM alignments not found by IBM Model 4.

2004; Liang et al., 2006; Ganchev et al., 2008), which aim to achieve symmetric word alignment during training, though good in many cases, are not designed to tackle highly inflected languages.

Our work differs from these by taking a middle road. Instead of modifying the alignment algorithm directly, we preprocess asymmetric alignments to improve the input to the symmetrizing process later. Also, our approach does not make use of specific language resources, relying only on unsupervised morphological analysis.

3 A Case for Morpheme Alignment

The notion that morpheme based alignment would be useful in highly inflected languages is intuitive. Morphological inflections might indicate tense, gender or number that manifest as separate words in largely uninflected languages. Capturing these subword alignments can yield better word alignments that otherwise would be missed.

Let us make this idea concrete with a case study of the benefits of morpheme based alignment. We show the intersecting alignments of an actual English (source) $\rightarrow$ Finnish (target) sentence pair in Figure 1, where (a) word-level and (b) morpheme-level alignments are shown. The morpheme-level alignment is produced by automatically segmenting words into morphemes and running IBM Model 4 on the resulting token stream.

Intersection links (*i.e.*, common to both direct and inverse alignments) play an important role in creating the final alignment (Och and Ney, 2004). While there are several heuristics used in the symmetrizing process, the *grow-diag(onal)* process is common and prevalent in many SMT systems, such as Moses (Koehn et al., 2007). In the grow-diag process, intersection links are used as seeds to find other new alignments within their neighborhood. The process continues iteratively, until no further links can be added.

In our example, the morpheme-level intersection alignment is better as it has no misalignments and adds new alignments. However it misses some key links. In particular, the alignments of closed-class morphemes (CCMs; later formally defined) as indicated by the dotted lines in (b) are overlooked in the IBM Model 4 alignment. This difficulty in aligning CCMs is due to:

1. Occurrences of *garbage-collector* words (Moore, 2004) that attract CCMs to align to them. Examples of such links in (b) are 1–23 or 11–21 with the occurrences of rare words `adjourn+`$_{11}$ and `avatuksi`$_{23}$. We further characterize such errors in Section 6.1.

2. Ambiguity among CCMs of the same surface that causes incorrect matchings. In (b), we observe multiple occurrence of `the` and `n` on the source and target sides respectively. While the link 8–6 is correct, 5–4 is not as $\mathtt{i}_1$ should be aligned to $\mathtt{n}_4$ instead. To resolve such ambiguity, context information should be considered as detailed in Section 4.3.

The fact that rare words and multiple affixes often occur in highly inflected languages exacerbates this problem, motivating our focus on improving CCM alignment. Furthermore, having access to the correct CCM alignments as illustrated

in Figure 1 guides the grow-diag process in finding the remaining correct alignments. For example, the addition of CCM links i$_1$–n$_4$ and d$_4$–lle$_{21}$ helps to identify declare$_2$–julist$_2$ and resume$_3$–avatuksi$_{23}$ as admissible alignments, which would otherwise be missed.

4 Methodology

Our idea is to enrich the standard IBM Model 4 alignment by modeling closed-class morphemes (CCMs) more carefully using global statistics and context. We realize our idea by proposing a four-step method. First, we take the input parallel corpus and convert it into morphemes before training the IBM Model 4 morpheme alignment. Second, from the morpheme alignment, we induce automatically bilingual CCM pairs. The core of our approach is in the third and fourth steps. In Step 3, we construct a *CCM alignment model*, and apply it on the segmented input corpus to obtain an automatic CCM alignment. Finally, in Step 4, we incorporate the CCM alignment into the symmetrizing process via our *modified grow-diag process*.

4.1 Step 1: Morphological Analysis

The first step presupposes morphologically segmented input to compute the IBM Model 4 morpheme alignment. Following Virpioja et al. (2007), we use *Morfessor*, an unsupervised analyzer which learns morphological segmentation from raw tokenized text (Creutz and Lagus, 2007).

The tool segments input words into labeled morphemes: PRE (prefix), STM (stem), and SUF (suffix). Multiple affixes can be proposed for each word; word compounding is allowed as well, e.g., uncarefully is analyzed as un/PRE+ care/STM+ ful/SUF+ ly/SUF. We append a "+" sign to each nonfinal tag to distinguish word-internal morphemes from word-final ones, e.g., "x/STM" and "x/STM+" are considered different tokens. The "+" annotation enables the restoration of the original words, a key point to enforce word boundary constraints in our work later.

4.2 Step 2: Bilingual CCM Pairs

We observe that low and highly inflected languages, while intrinsically different, share more in common at the morpheme level. The many-to-one relationships among words on both sides is often captured better by one-to-one correspondences among morphemes. We wish to model such bilingual correspondence in terms of closed-class morphemes (CCM), similar to Nguyen and Vogel (2008)'s work that removes nonaligned affixes during the alignment process. Let us now formally define CCM and an associative measure to gauge such correspondence.

en	fi	en	fi	en	fi
the$_1$	-n$_1^\dagger$	in$_6$	-ssa$_{15}^\ddagger$	me$_{166}$	-ni$_{60}^\ddagger$
-s$_2$	-t$_9^\ddagger$	is$_7$	on$_2^\ddagger$	me$_{166}$	minun$_{282}^\dagger$
to$_3$	-ä$_6$	that$_8$	että$_7^\ddagger$	why$_{168}$	siksi$_{187}^\ddagger$
to$_3$	maan$_{91}$	that$_8$	ettei$_{283}^\ddagger$	view$_{172}$	mieltä$_{162}^\dagger$
of$_4$	-a$_4$	we$_{10}$	-mme$_{10}^\ddagger$	still$_{181}$	vielä$_{108}^\ddagger$
of$_4$	-en$_5^\dagger$	we$_{10}$	meidän$_{52}^\dagger$	where$_{183}$	jossa$_{209}^\ddagger$
of$_4$	-sta$_{19}^\dagger$	we$_{10}$	me$_{113}^\ddagger$	same$_{186}$	samaa$_{334}^\ddagger$
and$_5$	ja$_3^\ddagger$	we$_{10}$	emme$_{123}$	he$_{187}$	hän$_{184}^\ddagger$
and$_5$	sekä$_{122}^\ddagger$	we$_{10}$	meillä$_{231}^\dagger$	good$_{189}$	hyvä$_{321}^\ddagger$
and$_5$	eikä$_{203}$	...	...	over-$_{408}$	yli-$_{391}^\ddagger$

Table 1: **English(en)-Finnish(fi) Bilingual CCM pairs** (N=128). Shown are the top 19 and last 10 of 168 bilingual CCM pairs extracted. Subscript i indicates the i^{th} most frequent morpheme in each language. ‡ marks exact correspondence linguistically, whereas † suggests rough correspondence w.r.t http://en.wiktionary.org/wiki/.

Definition 1. *Closed-class Morphemes (CCM)* are a *fixed set of stems and affixes* that exhibit grammatical functions just like closed-class words. In highly inflected languages, we observe that grammatical meanings may be encoded in morphological stems and affixes, rather than separate words. While we cannot formally identify valid CCMs in a language-independent way (as by definition they manifest language-dependent grammatical functions), we can devise a good approximation. Following Setiawan et al. (2007), we induce the set of CCMs for a language as the top N frequent stems together with all affixes[1].

Definition 2. *Bilingual Normalized PMI (biPMI)* is the averaged normalized PMI computed on the asymmetric morpheme alignments. Here, normalized PMI (Bouma, 2009), known to be less biased towards low-frequency data, is defined as: $nPMI(x,y) = \ln \frac{p(x,y)}{p(x)p(y)} / -\ln p(x,y)$, where $p(x)$, $p(y)$, and $p(x,y)$ follow definitions in the standard PMI formula. In our case, we only

[1]Note that we employ length and vowel sequence heuristics to filter out corpus-specific morphemes.

compute the scores for x, y being morphemes frequently aligned in both asymmetric alignments.

Given these definitions, we now consider a pair of source and target CCMs related and termed a **bilingual CCM pair** (CCM pair, for short) if they exhibit positive correlation in their occurrences (*i.e.*, positive nPMI[2] and frequent cooccurrences).

We should note that relying on a hard threshold of N as in (Setiawan et al., 2007) is brittle as the CCM set varies in sizes across languages. Our method is superior in the use of N as a starting point only; the bilingual correspondence of the two languages will ascertain the final CCM sets.

Take for example the *en* and *fi* CCM sets with 154 and 214 morphemes initially (each consisting of N=128 stems). As morphemes not having their counterparts in the other language are spurious, we remove them by retaining only those in the CCM pairs. This effectively reduces the respective sizes to 91 and 114. At the same time, these final CCMs cover a much larger range of top frequent morphemes than N, up to 408 *en* and 391 *fi* morphemes, as evidenced in Table 1.

4.3 Step 3: The CCM Alignment Model

The goal of this model is to predict when appearances of a CCM pair should be deemed as linking.

With an identified set of CCM pairs, we know when source and target morphemes correspond. However, in a sentence pair there can be many instances of both the source and target morphemes. In our example, the `the`–`n` pair corresponds to definite nouns; there are two `the` and three `-n` instances, yielding 2×3=6 possible links.

Deciding which instances are aligned is a decision problem. To solve this, we inspect the IBM Model 4 morpheme alignment to construct a CCM alignment model. The CCM model labels whether an instance of a CCM pair is deemed semantically related (linked). We cast the modeling problem as supervised learning, where we choose a maximum entropy (ME) formulation (Berger et al., 1996).

We first discuss sample selection from the IBM Model 4 morpheme alignment, and then give details on the features extracted. The processes described below are done per sentence pair with f_1^m, e_1^n and U denoting the source, target sentences and the union alignments, respectively.

Class labels. We base this on the initial IBM Model 4 alignment to label each CCM pair instance as a positive or negative example: *Positive examples* are simply CCM pairs in U. To be precise, links j–i in U are positive examples if f_j–e_i is a CCM pair. To find *negative examples*, we inventory other potential links that share the same lexical items with a positive one. That is, a link j'–i' not in U is a negative example, if a positive link j–i such that $f_j = f'_j$ and $e_i = e'_i$ exists.

We stress that our collection of positive examples contains high-precision but low-recall IBM Model 4 links, which connect the reliable CCM pairs identified before. The model then generalizes from these samples to detect incorrect CCM links and to recover the correct ones, enhancing recall. We later detail this process in §4.4.

Feature Set. Given a CCM pair instance, we construct three feature types: lexical, monolingual, and bilingual (See Table 2). These features capture the global statistics and contexts of CCM pairs to decide if they are true alignment links.

• **Lexical features** reflect the tendency of the CCM pair being aligned to themselves. We use *biPMI*, which aggregates the global alignment statistics, to determine how likely source and target CCMs are associated with each other.

• **Monolingual context features** measure the association among tokens of the same language, capturing what other stems and affixes co-occur with the source/target CCM:

1. within the same word (*intra*). The aim is to disambiguate affixes as necessary in highly inflected languages where same stems could generate different roles or meanings.
2. outside the CCM's word boundary (*inter*). This potentially capture cues such as tense, or number agreement. For example, in English, the 3sg agreement marker on verbs `-s` often co-occurs with nearby pronouns e.g., `he`, `she`, `it`; whereas the same marker on nouns (`-s`), often appears with plural determiners e.g., `these`, `those`, `many`.

To accomplish this, we compute two monolingual nPMI scores in the same spirit as biPMI, but using the morphologically segmented input from

[2] *nPMI* has a bounded range of $[-1, 1]$ with values 1 and 0 indicating perfect positive and no correlation, respectively.

Feature Description	Examples
Lexical — *biPMI*: None $[-1, 0]$, Low $(0, 1/3]$, Medium $(1/3, 2/3]$, High $(2/3, 1]$	pmi_{d-lle}=Low
Monolingual Context — Capture morpheme cooccurrence with the src/tgt CCM	
Intra – **W**ithin the same word	srcW_{d-lle}=resume, tgtW_{d-lle}=en, tgtW_{d-lle}=uude
Inter – To the **L**eft & **R**ight, in different words	srcL_{d-lle}=i, srcR_{d-lle}=the, tgtR_{d-lle}=avatuksi
Bilingual context — Capture neighbor links' cooccurrence with the CCM pair link	
bi0 – Most descriptive, capturing in terms of surface forms only → maybe sparse	bi0_{d-lle}=resume–avatuksi
bi1 – Generalizes morphemes into relative locations (**L**eft, **W**ithin, **R**ight)	bi1_{d-lle}=W–avatuksi, bi1_{d-lle}=resume–R
bi2 – Most general, coupling token types (**C**lose, **O**pen) /w relative positions	bi2_{d-lle}=O–WR

Table 2: **Maximum entropy feature set.** Shown are feature types, descriptions and examples. Most examples are given for the alignment $\texttt{d}_4\texttt{–lle+}_{21}$ of the same running example in §3. Note that we only partially list the bilingual context features.

each language separately. Two morphemes are "linked" if within a context window of w_c words.

• **Bilingual context features** model cross-lingual reordering, capturing the relationships between the CCM pair link and its neighbor[3] links. Consider a simple translation between an English phrase of the form `we` ⟨`verb`⟩ and the Finnish one ⟨`verb`⟩ `–mme`, where `–mme` is the 1pl verb marker. We aim to capture movements such as "the open-class morphemes on the right of `we` and on the left of `–mme` are often aligned". These will function as evidence for the ME learner to align the CCM pair (`we`, `–mme`). We encode the bilingual context at three different granularities, from most specific to most general ones (cf Table 2).

4.4 Step 4: Incorporate CCM Alignment

At test time, we apply the trained CCM alignment model to all CCM pairs occurring in each sentence pair to find CCM links. On our running example in Figure 1, the CCM classifier tests 17 CCM pairs, identifying 6 positive CCM links of which 4 are true positives (dotted lines in (b)).

Though mostly correct, we note that some of the predicted links conflict: ($\texttt{d}_4\texttt{–lle}_{21}$, $\texttt{ed}_{12}\texttt{–neen}_{17}$ and $\texttt{ed}_{12}\texttt{–lle}_{21}$) share alignment endpoints. Such sharing in CCM alignments is rare and we believe should be disallowed. This motivates us to resolve all CCM link conflicts before incorporating them into the symmetrizing process.

Resolving link conflicts. As CCM pairs are classified independently, they possess classification probabilities which we use as evidence to resolve the conflicts. In our example, the classification probabilities for ($\texttt{d}_4\texttt{–lle}_{21}$, $\texttt{ed}_{12}\texttt{–neen}_{17}$, $\texttt{ed}_{12}\texttt{–lle}_{21}$) are $(0.99, 0.93, 0.79)$ respectively.

We use a simple, "best-first" greedy approach to determine which links are kept and which are dropped to satisfy our assumption. In our case, we pick the most confident link, $\texttt{d}_4\texttt{–lle}_{21}$ with probability 0.99. This precludes the incorrect link, $\texttt{ed}_{12}\texttt{–lle}_{21}$, but admits the other correct one $\texttt{ed}_{12}\texttt{–neen}_{17}$, probability 0.93. As a result, this resolution successfully removes the incorrect link.

Modifying grow-diag. We incorporate the set of conflict-resolved CCM links into the grow-diag process. This step modifies the input alignments as well as the growing process. U and I denote the IBM Model 4 union and intersection alignments.

In our view, the resolved CCM links can serve as a quality mark to "upgrade" links before input into the grow-diag process. We upgrade resolved CCM links: (a) those $\in U \rightarrow$ part of I, treating them as alignment seeds; (b) those $\notin U \rightarrow$ part of U, using them for exploration and growing. To reduce spurious alignments, we discarded links in U that conflict with the resolved CCM links.

In the usual grow-diag, links immediately adjacent to a seed link l are considered candidates to be appended into the alignment seeds. While suitable for word-based alignment, we believe it is too small a context when the input are morphemes.

For morpheme alignment, the candidate context makes more sense in terms of word units. We thus *enforce word boundaries* in our modified grow-diag. We derive word boundaries for end points in l using the morphological tags and the "+" word-end marker mentioned in §4.1. Using such boundaries, we can then extend the grow-diag to consider candidate links within a neighborhood of w_g words; hence, enhancing the candidate coverage.

5 Experiments

We use English-Finnish and English-Hungarian data from past shared tasks (WPT05 and WMT09)

[3] Within a context window of w_c words as in monolingual.

to validate our approach. Both Finnish and Hungarian are highly inflected languages, with numerous verbal and nominal cases, exhibiting agreement. Dataset statistics are given in Table 3.

	en-fi	#	en-hu	#
Train	Europarl-v1	714K	Europarl-v4	1,510K
LM	Europarl-v1-fi	714K	News-hu	4,209K
Dev	wpt05-dev	2000	news-dev2009	2051
Test	wpt05-test	2000	news-test2009	3027

Table 3: **Dataset Statistics:** the numbers of parallel sentences for training, LM training, development and test sets.

We use the Moses SMT framework for our work, creating both our CCM-based systems and the baselines. In all systems built, we obtain the IBM Model 4 alignment via GIZA++ (Och and Ney, 2003). Results are reported using case-insensitive BLEU (Papineni et al., 2001).

Baselines. We build two SMT baselines:

w-system: This is a standard phrase-based SMT, which operates at the word level. The system extracts phrases of maximum length 7 words, and uses a 4-gram word-based LM.

w_m-system: This baseline works at the word level just like the w-system, but differs at the alignment stage. Specifically, input to the IBM Model 4 training is the morpheme-level corpus, segmented by *Morfessor* and augmented with "+" to provide word-boundary information (§4.1). Using such information, we constrain the alignment symmetrization to extract phrase pairs of 7 words or less in length. The morpheme-based phrase table is then mapped back to word forms. The process continues identically as in the w-system.

CCM-based systems. Our CCM-based systems are similar in spirit to the w_m system: train at the morpheme, but decode at the word level. We further enhance the w_m-system at the alignment stage. First, we train our CCM model based on the initial IBM Model 4 morpheme alignment, and apply it to the morpheme corpus to obtain CCM alignment, which are input to our modified grow-diag process. The CCM approach defines the setting of three parameters: $\langle N, w_c, w_g \rangle$ (Section 4). Due to our resource constraints, we set N=128, similar to (Setiawan et al., 2007), and w_c=1 experimentally. We only focus on the choice of w_g, testing w_g=$\{1, 2\}$ to explore the effect of enforcing word boundaries in the grow-diag process.

5.1 English-Finnish results

We test the translation quality of both directions (*en-fi*) and (*fi-en*). We present results in Table 4 for 7 systems, including: our baselines, three CCM-based systems with word-boundary knowledge w_g=$\{0, 1, 2\}$ and two w_m-systems w_g=$\{1, 2\}$.

Results in Table 4 show that our CCM approach effectively improves the performance. Compared to the w_m-system, it chalks up a gain of 0.46 BLEU points for *en-fi*, and a larger improvement of 0.93 points for the easier, reverse direction.

Further using word boundary knowledge in our modified grow-diag process demonstrates that the additional flexibility consistently enhances BLEU for $w_g = 1, 2$. We achieve the best performance at $w_g = 2$ with improvements of 0.67 and 1.22 BLEU points for *en-fi* and *fi-en*, respectively.

	en-fi	fi-en
w-system	14.58	23.56
w_m-system	14.47	22.89
w_m-system + CCM	$14.93_{+0.46}$	$23.82_{+0.93}$
w_m-system + CCM + $w_g = 1$	15.01	23.95
w_m-system + CCM + $w_g = 2$	$\mathbf{15.14}_{+0.67}$	$\mathbf{24.11}_{+1.22}$
w_m-system + $w_g = 1$	14.44	22.92
w_m-system + $w_g = 2$	14.28	23.01
(Ganchev, 2008) - Base	14.72	22.78
(Ganchev, 2008) - Postcat	14.74	$23.43_{+0.65}$
(Yang, 2006) - Base	N/A	22.0
(Yang, 2006) - Backoff	N/A	$22.3_{+0.3}$

Table 4: **English/Finnish results.** Shown are BLEU scores (in %) with subscripts indicating absolute improvements with respect to the w_m-system baseline.

Interestingly, employing the word boundary heuristic alone in the original grow-diag does not yield any improvement for *en-fi*, and even worsens as w_g is enlarged (as seen in Rows 6–7). There are only slight improvements for *fi-en* with larger w_g.This attests to the importance of combining the CCM model and the modified grow-diag process.

Our best system outperforms the w-system baseline by 0.56 BLEU points for *en-fi*, and yields an improvement of 0.55 points for *fi-en*.

Compared to works experimenting *en/fi* translation, we note the two prominent ones by Yang and Kirchhoff (2006) and recently by Ganchev et al. (2008). The former uses a simple back-off method experimenting only *fi-en*, yielding an improvement of 0.3 BLEU points. Work in the op-

posite direction (*en-fi*) is rare, with the latter paper extending the EM algorithm using posterior constraints, but showing no improvement; for *fi-en*, they demonstrate a gain of 0.65 points. Our CCM method compares favorably against both approaches, which use the same datasets as ours.

5.2 English-Hungarian results

To validate our CCM method as language-independent, we also perform preliminary experiments on *en-hu*. Table 5 shows the results using the same CCM setting and experimental schemes as in *en/fi*. An improvement of 0.35 BLEU points is shown using the CCM model. We further improve by 0.44 points with word boundary w_g=1, but performance degrades for the larger window. Due to time constraints, we leave experiments for the reverse, easier direction as future work. Though modest, the best improvement for *en-hu* is statistical significant at $p<0.01$ according to Collins' sign test (Collins et al., 2005).

System	BLEU
w-system	9.63
w_m-system	9.47
w_m-system + CCM	9.82 $_{+0.35}$
w_m-system + CCM + $w_g = 1$	**9.91** $_{+0.44}$
w_m-system + CCM + $w_g = 2$	9.87

Table 5: **English/Hungarian results.** Subscripts indicate absolute improvements with respect to the w_m-system.

We note that MT experiments for *en/hu* [4] are very limited, especially for the *en* to *hu* direction. Novák (2009) obtained an improvement of 0.22 BLEU with no distortion penalty; whereas Koehn and Haddow (2009) enhanced by 0.5 points using monotone-at-punctuation reordering, minimum Bayes risk and larger beam size decoding.

While not directly comparable in the exact settings, these systems share the same data source and splits similar to ours. In view of these community results, we conclude that our CCM model does perform competitively in the *en-hu* task, and indeed seems to be language independent.

6 Detailed Analysis

The macroscopic evaluation validates our approach as improving BLEU over both baselines, but how do the various components contribute? We first analyze the effects of Step 4 in producing the CCM alignment, and then step backward to examine the contribution of the different feature classes in Step 3 towards the ME model.

6.1 Quality of CCM alignment

To evaluate the quality of the predicted CCM alignment, we address the following questions:

Q1: What is the portion of CCM pairs being misaligned in the IBM Model 4 alignment?

Q2: How does the CCM alignment differ from the IBM Model 4 alignment?

Q3: To what extent do the new links introduced by our CCM model address Q1?

Given that we do not have linguistic expertise in Finnish or Hungarian, it is not possible to exhaustively list all misaligned CCM pairs in the IBM Model 4 alignment. As such, we need to find other form of approximation in order to address Q1.

We observe that correct links that do not exist in the original alignment could be entirely missing, or mistakenly aligned to neighboring words. With morpheme input, we can also classify mistakes with respect to intra- or inter-word errors. Figure 2 characterizes errors T_1, T_2 and T_3, each being a more severe error class than the previous. Focusing on e_i in the figure, links connecting e_i to $f_{j'}$ or $f_{j''}$ are deemed T_1 errors (misalignments happen on one side). A T_2 error aligns f''_j within the same word, while a T_3 error aligns it outside the current word but still within its neighborhood. This characterization is automatic, cheap and has the advantage of being language-independent.

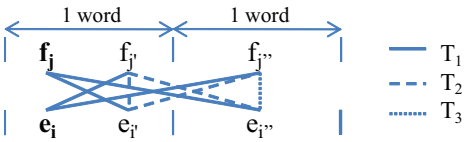


Figure 2: **Categorization of CCM missing links.** Given that a CCM pair link (f_j–e_i) is not present in the IBM Model 4, occurrences of any nearby link of the types $T_{[1-3]}$ can be construed as evidence of a potential misalignment.

Statistics in Table 6(ii) answers Q1, suggesting a fairly large number of missing CCM links: $3,418K$ for *en/fi* and $6,216K$ for *en/hu*, about 12.35% and 12.06% of the IBM Model 4 union alignment respectively. We see that T_1 errors con-

[4] Hungarian was used in the ACL shared task 2008, 2009.

stitute the majority, a reasonable reflection of the *garbage- collector*[5] effect discussed in Section 3.

	General (i)		Missing CCM links (ii)		
	en/fi	en/hu		en/fi	en/hu
Direct	17,632K	34,312K	T_1	2,215K	3,487K
Inverse	18,681K	34,676K	T_2	358K	690K
$D \cap I$	8,643K	17,441K	T_3	845K	2,039K
$D \cup I$	27,670K	51,547K	**Total**	**3,418K**	**6,216K**

Table 6: **IBM Model 4 alignment statistics.** (i) General statistics. (ii) Potentially missing CCM links.

Q2 is addressed by the last column in Table 7. Our CCM model produces about 11.98% (1,035K/8,643K) new CCM links as compared to the size of the IBM Model 4 intersection alignment for en/fi, and similarly, 9.52% for en/hu.

	Orig.	Resolved	I	U\I	New
en/fi	5,299K	3,433K	1065K	1,332K	**1,035K**
en/hu	9,425K	6,558K	2,752K	2,146K	**1,660K**

Table 7: **CCM vs IBM Model 4 alignments.** *Orig.* and *Resolved* give # CCM links predicted in Step 4 before and after resolving conflicts. Also shown are the number of resolved links present in the **I**ntersection, Union excluding I (**U\I**) of the IBM Model 4 alignment and **New** CCM links.

Lastly, figures in Table 8 answer Q3, revealing that for *en/fi*, 91.11% (943K/1,035K) of the new CCM links effectively cover the missing CCM alignments, recovering 27.59% (943K/3,418K) of all missing CCM links. Our modified *grow-diag* realizes a majority 76.56% (722K/943K) of these links in the final alignment.

We obtain similar results in the *en/hu* pair for link recovery, but a smaller percentage 22.59% (330K/1,461K) are realized through the modified symmetrization. This partially explains why improvements are modest for *en/hu*.

	New CCM Links (i)		Modified grow-diag (ii)	
	en/fi	en/hu	en/fi	en/hu
T_1	707K	1,002K	547K	228K
T_2	108K	146K	79K	22K
T_3	128K	313K	96K	80K
Total	**943K**	**1,461K**	**722K**	**330K**

Table 8: **Quality of the newly introduced CCM links**. Shown are # new CCM links addressing the three error types before (i) and after (ii) the modified grow-diag process.

6.2 Contributions of ME Feature Classes

We also evaluate the effectiveness the ME features individually through ablation tests. For brevity, we only examine the more difficult translation direction, *en* to *fi*. Results in Table 9 suggest that all our features are effective, and that removing any feature class degrades performance. Balancing specificity and generality, `bi1` is the most influential feature in the bilingual context group. For monolingual context, `inter`, which captures larger monolingual context, outperforms `intra`. The most important feature overall is `pmi`, which captures global alignment preferences. As feature groups, bilingual and monolingual context features are important sources of information, as removing them drastically decreases system performance by 0.23 and 0.16 BLEU, respectively.

System			BLEU
all (w_m-system+CCM)			14.93
−bi2	14.90	−intra	14.89
−bi1	$14.84^{*}_{-0.09}$	−pmi	$14.81^{*}_{-0.12}$
−bi0	14.89	−bi{2/1/0}	$14.70^{*}_{-0.23}$
−inter	14.85	−in{ter/tra}	$14.77^{*}_{-0.16}$

Table 9: **ME feature ablation tests for English-Finnish experiments.** $*$ mark results statistically significant at $p < 0.05$, differences are subscripted.

7 Conclusion and Future Work

In this work, we have proposed a language-independent model that addresses morpheme alignment problems involving highly inflected languages. Our method is unsupervised, requiring no language specific information or resources, yet its improvement on BLEU is comparable to much semantically richer, language-specific work. As our approach deals only with input word alignment, any downstream modifications of the translation model also benefit.

As alignment is a central focus in this work, we plan to extend our work over different and multiple input alignments. We also feel that better methods for the incorporation of CCM alignments is an area for improvement. In the en/hu pair, a large proportion of discovered CCM links are discarded, in favor of spurious links from the union alignment. Automatic estimation of the correctness of our CCM alignments may improve end translation quality over our heuristic method.

[5] E.g., e_i prefers f'_j or f''_j (garbage collectors) over f_j.

References

Berger, Adam L., Stephen D. Della Pietra, and Vincent J. D. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Bouma, Gerlof. 2009. Normalized (pointwise) mutual information in collocation extraction. In *Proceedings of the Biennial GSCL Conference*, Tübingen, Gunter Narr Verlag.

Cherry, Colin and Dekang Lin. 2007. Inversion transduction grammar for joint phrasal translation modeling. In *SSST*.

Collins, Michael, Philipp Koehn, and Ivona Kučerová. 2005. Clause restructuring for statistical machine translation. In *ACL*.

Creutz, Mathias and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Trans. Speech Lang. Process.*, 4(1):3.

DeNero, John and Dan Klein. 2007. Tailoring word alignments to syntactic machine translation. In *ACL*.

Ganchev, Kuzman, João V. Graça, and Ben Taskar. 2008. Better alignments = better translations? In *ACL-HLT*.

Goldwater, Sharon and David McClosky. 2005. Improving statistical mt through morphological analysis. In *HLT*.

Haghighi, Aria, John Blitzer, John DeNero, and Dan Klein. 2009. Better word alignments with supervised itg models. In *ACL*.

Koehn, Philipp and Barry Haddow. 2009. Edinburgh's submission to all tracks of the WMT2009 shared task with reordering and speed improvements to Moses. In *EACL*.

Koehn, Philipp, Hieu Hoang, Alexandra Birch Mayne, Christopher Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL, Demonstration Session*.

Lee, Young-Suk. 2004. Morphological analysis for statistical machine translation. In *HLT-NAACL*.

Liang, Percy, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *HLT-NAACL*.

Matusov, Evgeny, Richard Zens, and Hermann Ney. 2004. Symmetric word alignments for statistical machine translation. In *COLING*.

Moore, Robert C. 2004. Improving IBM word-alignment model 1. In *ACL*.

Nguyen, Thuy Linh and Stephan Vogel. 2008. Context-based Arabic morphological analysis for machine translation. In *CoNLL*.

Novák, Attila. 2009. MorphoLogic's submission for the WMT 2009 shared task. In *EACL*.

Och, Franz Josef and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Och, Franz Josef and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: a method for automatic evaluation of machine translation. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Morristown, NJ, USA. Association for Computational Linguistics.

Setiawan, Hendra, Min-Yen Kan, and Haizhou Li. 2007. Ordering phrases with function words. In *ACL*.

Virpioja, Sami, Jaakko J. Vyrynen, Mathias Creutz, and Markus Sadeniemi. 2007. Morphology-aware statistical machine translation based on morphs induced in an unsupervised manner. In *MT Summit XI*.

Yang, Mei and Katrin Kirchhoff. 2006. Phrase-based backoff models for machine translation of highly inflected languages. In *EACL*.

Zhang, Hao and Daniel Gildea. 2005. Stochastic lexicalized inversion transduction grammar for alignment. In *ACL*.

Zhang, Hao, Chris Quirk, Robert C. Moore, and Daniel Gildea. 2008. Bayesian learning of non-compositional phrases with synchronous parsing. In *ACL-HLT*.

Automatic analysis of semantic similarity in comparable text through syntactic tree matching

Erwin Marsi
TiCC, Tilburg University
e.c.marsi@uvt.nl

Emiel Krahmer
TiCC, Tilburg University
e.j .krahmer@uvt.nl

Abstract

We propose to analyse semantic similarity in comparable text by matching syntactic trees and labeling the alignments according to one of five semantic similarity relations. We present a Memory-based Graph Matcher (MBGM) that performs both tasks simultaneously as a combination of exhaustive pairwise classification using a memory-based learner, followed by global optimization of the alignments using a combinatorial optimization algorithm. The method is evaluated on a monolingual treebank consisting of comparable Dutch news texts. Results show that it performs substantially above the baseline and close to the human reference.

1 Introduction

Natural languages allow us to express essentially the same underlying meaning as many alternative surface forms. In other words, there are often many similar ways to say the same thing. This characteristic poses a problem for many natural language processing applications. Automatic summarizers, for example, typically rank sentences according to their informativity and then extract the top n sentences, depending on the required compression rate. Although the sentences are essentially treated as independent of each other, they typically are not. Extracted sentences may have substantial semantic overlap, resulting in unintended redundancy in the summaries. This is particularly problematic in the case of multi-document summarization, where sentences extracted from related documents are very likely to express similar information in different ways (Radev and McKeown, 1998). Therefore, if semantic similarity between sentences could be detected automatically, this would certainly help to avoid redundancy in summaries.

Similar arguments can be made for many other NLP applications. Automatic duplicate and plagiarism detection beyond obvious string overlap requires recognition of semantic similarity. Automatic question-answering systems may benefit from clustering semantically similar candidate answers. Intelligent document merging software, which supports a minimal but lossless merge of several revisions of the same text, must handle cases of paraphrasing, restructuring, compression, etc. Recognizing textual entailments (Dagan et al., 2005) could arguably be seen as a specific instance of detecting semantic similarity.

In addition to merely *detecting* semantic similarity, we can ask to what extent two expressions share meaning. For instance, the meaning of one sentence can be fully contained in that of another, the meaning of one sentence can overlap only partly with that of another, etc. This requires an *analysis* of the semantic similarity between a pair of expressions. Like detection, automatic analysis of semantic similarity can play an important role in NLP applications. To return to the case of multi-document summarization, analysing the semantic similarity between sentences extracted from different documents provides the basis for *sentence fusion*, a process where a new sentence is generated that conveys all common information from both sentences without introducing redundancy (Barzilay and McKeown, 2005; Marsi and Krahmer, 2005b).

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 752–760,
Beijing, August 2010

Analysis of semantic similarity can be approached from different angles. A basic approach is to use string similarity measures such as the Levenshtein distance or the Jaccard similarity coefficient. Although cheap and fast, this fails to account for less obvious cases such as synonyms or syntactic paraphrasing. At the other extreme, we can perform a deep semantic analysis of two expressions and rely on formal reasoning to derive a logical relation between them. This approach suffers from issues with coverage and robustness commonly associated with deep linguistic processing. We therefore think that the middle ground between these two extremes offers the best option. In this paper we present a new method for analysing semantic similarity in comparable text. It relies on a combination of morphological and syntactic analysis, lexical resources such as word nets, and machine learning from examples. We propose to analyse semantic similarity between sentences by aligning their syntax trees, where each node is matched to the most similar node in the other tree (if any). In addition, we label these alignments according to the type of similarity relation that holds between the aligned phrases. The labeling supports further processing. For instance, Marsi & Krahmer (2005b; 2008) describe how to generate different types of sentence fusions on the basis of this relation labeling.

In the next Section we provide a more formal definition of the task of matching syntactic trees and labeling alignments, followed by a discusion of related work in Section 3. Section 4 describes a parallel, monolingual treebank used for developing and testing our approach. In Section 5 we propose a new algorithm for simultaneous node alignment and relation labeling. The results of several evaluation experiments are presented in Section 6. We finish with a conclusion.

2 Problem statement

Aligning a pair of similar syntactic trees is the process of pairing those nodes that are most similar. More formally: let v be a node in the syntactic tree T of sentence S and v' a node in the syntactic tree T' of sentence S'. A *labeled node alignment* is a tuple $< v, v', r >$ where r is a label from a set of relations. A *labeled tree alignment* is a set of labeled node alignments. A *labeled tree matching* is a tree alignment in which each node is aligned to at most one other node.

For each node v, its terminal *yield* $\text{STR}(v)$ is defined as the sequence of all terminal nodes reachable from v (i.e., a substring of sentence S). Aligning node v to v' with label r indicates that relation r holds between their yields $\text{STR}(v)$ and $\text{STR}(v')$. We label alignments according to a small set of *semantic similarity relations*. As an example, consider the following Dutch sentences:

(1) a. Dagelijks koffie vermindert risico op
Daily coffee diminishes risk on
Alzheimer en Dementie.
Alzheimer and Dementia.

b. Drie koppen koffie per dag reduceert
Three cups coffee a day reduces
kans op Parkinson en Dementie.
chance on Parkinson and Dementia.

The corresponding syntax trees and their (partial) alignment is shown in Figure 1. We distinguish the following five mutually exclusive similarity relations:

1. v **equals** v' iff lower-cased $\text{STR}(v)$ and lower-cased $\text{STR}(v')$ are identical – example: *Dementia* equals *Dementia*;
2. v **restates** v' iff $\text{STR}(v)$ is a proper paraphrase of $\text{STR}(v')$ – example: *diminishes* restates *reduces*;
3. v **generalizes** v' iff $\text{STR}(v)$ is more general than $\text{STR}(v')$ – example: *daily coffee* generalizes *three cups of coffee a day*;
4. v **specifies** v' iff $\text{STR}(v)$ is more specific than $\text{STR}(v')$ – example: *three cups of coffee a day* specifies *dailly coffee*;
5. v **intersects** v' iff $\text{STR}(v)$ and $\text{STR}(v')$ share meaning, but each also contains unique information not expressed in the other – example: *Alzheimer and Dementia* intersects *Parkinson and Dementia*.

Our interpretation of these relations is one of common sense rather than strict logic, akin to the definition of entailment employed in the RTE challenge (Dagan et al., 2005). Note also that relations are prioritized: *equals* takes precedence

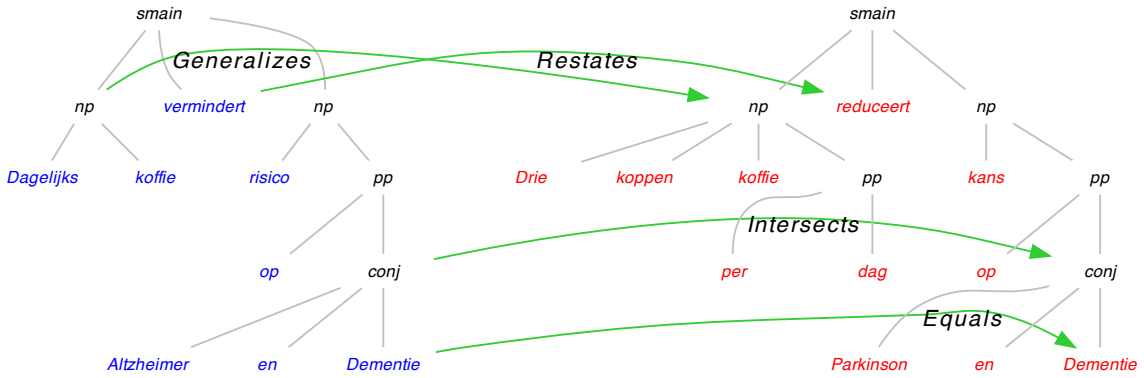


Figure 1: Example of two aligned and labeled syntactic trees. For expository reasons the alignment is not exhaustive.

over *restates*, etc. Furthermore, *equals*, *restates* and *intersects* are symmetrical, whereas *generalizes* is the inverse of *specifies*. Finally, nodes containing unique information, such as *Alzheimer* and *Parkinson*, remain unaligned.

3 Related work

Many syntax-based approaches to machine translation rely on bilingual treebanks to extract transfer rules or train statistical translation models. In order to build bilingual treebanks a number of methods for automatic tree alignment have been developed, e.g., (Gildea, 2003; Groves et al., 2004; Tinsley et al., 2007; Lavie et al., 2008). Most related to our approach is the work on discriminative tree alignment by Tiedemann & Kotzé (2009). However, these algorithms assume that source and target sentences express the same information (i.e. *parallel* text) and cannot cope with comparable text where parts may remain unaligned. See (MacCartney et al., 2008) for further arguments and empirical evidence that MT alignment algorithms are not suitable for aligning parallel monolingual text.

MacCartney, Galley, and Manning (2008) describe a system for monolingual phrase alignment based on supervised learning which also exploits external resources for knowledge of semantic relatedness. In contrast to our work, they do not use syntactic trees or similarity relation labels. Partly similar semantic relations are used in (MacCartney and Manning, 2008) for modeling semantic containment and exclusion in natural language inference. Marsi & Krahmer (2005a) is closely related to our work, but follows a more complicated method: first a dynamic programming-based tree alignment algorithm is applied, followed by a classification of similarity relations using a supervised-classifier. Other differences are that their data set is much smaller and consists of parallel rather than comparable text. A major drawback of this algorithmic approach it that it cannot cope with crossing alignments. We are not aware of other work that combines alignment with semantic relation labeling, or algorithms which perform both tasks simultaneously.

4 Data collection

For developing our alignment algorithm we use the DAESO corpus[1]. This is a Dutch parallel monolingual treebank of 1 million words, half of which were manually annotated. The corpus consists of pairs of sentences with different levels of semantic overlap, ranging from high (different Dutch translations of books from Darwin, Montaigne and Saint-Exupéry) to low (different press releases from the two main news agencies in The Netherlands, ANP and NOVUM). For this paper, we concentrate on the latter part of the DAESO corpus, where the proportion of Equals and Restates is relatively low. This corpus segment consists of 8,248 pairs of sentences, containing 162,361 tokens (ignoring punctuation). All sentences were tokenized and tagged, and subsequently parsed by the Alpino dependency parser for Dutch (Bouma et al., 2001). Two annota-

[1] http://daeso.uvt.nl

		Alignment:	Labeling:						
			Eq:	Re:	Spec:	Gen:	Int:	Macro:	Micro:
Words:	F:	95.38	95.48	58.50	65.81	65.00	25.85	62.11	88.72
	SD:	2.16	2.69	7.63	13.05	11.25	18.74		
Full trees:	F:	88.31	95.83	71.38	60.21	66.71	62.67	71.36	81.92
	SD:	1.15	2.27	3.77	7.63	8.17	6.14		

Table 1: Average F-scores (in percentages, with Standard Deviations) for the six human annotators on alignment and semantic relation labeling, for words and for full syntactic trees.

tors determined which sentences in the comparable news reports contained semantic overlap. Six other annotators produced manual alignments of words and phrases in matched sentence pairs, which resulted in 86,227 aligned pairs of nodes.

A small sample of 10 similar press releases comprising a total of 48 sentence pairs was independently annotated by all six annotators to determine inter-annotator agreement. We used precision, recall and F-score on alignment. To calculate these scores for relation labeling, we simply restrict the set of alignments to those labeled with a particular relation, ignoring all others. Likewise, we restrict these sets to terminal node alignments in order to get scores on word alignment.

Given the six annotations $A_1, \ldots, A_6$, we repeatedly took one as the $True$ annotation against which the five other annotations were evaluated. We then computed the average scores over these $6 * 5 = 30$ scores (note that with this procedure, precision, recall and F score end up being equal). Table 1 summarizes the results, both for word alignments and for full syntactic tree alignment. It can be seen that for alignment of words an average F-score of over 95 % was obtained, while alignment for full syntactic trees results in an F-score of 88%. For relation labeling, the scores differed per relation, as is to be expected: the average F-score for Equals was over 95% for both word and full tree alignment[2], and for the other relations average F-scores between 0.6 and 0.7 were obtained. The exception to note is Intersects on word level, which only occurred a few times according to a few of the annotators. The macro and micro (weighted) F-score averages on labeled alignment are 62.11% and 88.72% for words, and 71.36% and 81.92% for full syntactic trees.

[2]At first sight, it may seem that labeling Equals is a trivial and deterministic task, for which the F-score should always be close to 100%. However, the same word may occur multiple times in the source or target sentences, which introduces ambiguity. This frequently occurs with function words such as determiners and prepositions. Moreover, choosing among several equivalent Equals alignments may sometimes involve a somewhat arbitrary decision. This situation arises, for instance, when a proper noun is mentioned just once in the source sentence but twice in the target sentence.

5 Memory-based Graph Matcher

In order to automatically perform the alignment and labeling tasks described in Section 2, we cast these tasks simultaneously as a combination of exhaustive pairwise classification using a supervised machine learning algorithm, followed by global optimization of the alignments using a combinatorial optimization algorithm. Input to the tree matching algorithm is a pair of syntactic trees consisting of a source tree T_s and a target tree T_t.

Step 1: Feature extraction For each possible pairing of a source node n_s in tree T_s and a target node n_t in tree T_t, create an instance consisting of feature values extracted from the input trees. Features can represent properties of individual nodes, e.g. the category of the source node is NP, or relations between nodes, e.g. source and target node share the same part-of-speech.

Step 2: Classification A generic supervised classifier is used to predict a class label for each instance. The class is either one of the semantic similarity relations or the special class *none*, which is interpreted as *no alignment*. Our implementation employs the memory-based learner TiMBL (Daelemans et al., 2009), a freely available, efficient and enhanced implementation of k-nearest neighbour classification. The classifier is trained on instances derived according to Step 1 from a parallel treebank of aligned and labeled syntactic trees.

Step 3: Weighting Associate a cost with each prediction so that high costs indicate low confidence in the predicted class and vice versa. We use the normalized entropy of the class labels in the set of nearest neighbours (H) defined as

$$H = -\frac{\sum_{c \in C} p(c)\ log_2\, p(c)}{log_2 |C|} \quad (1)$$

where C is the set of class labels encountered in the set of nearest neighbours (i.e., a subset of the five relations plus *none*), and $p(c)$ is the probability of class c, which is simply the proportion of instances with class label c in the set of nearest neighbours. Intuitively this means that the cost is zero if all nearest neighbours are of the same class, whereas the cost goes to 1 if the nearest neighbours are equally distributed over all possible classes.

Step 4: Matching The classification step will usually give rise to one-to-many alignment of nodes. In order to reduce this to just one-to-one alignments, we search for a node matching which minimizes the sum of costs over all alignments. This is a well-known problem in combinatorial optimization known as the *Assignment Problem*. The equivalent in graph-theoretical terms is a *minimum weighted bipartite graph matching*. This problem can be solved in polynomial time ($O(n^3)$) using e.g., the *Hungarian algorithm* (Kuhn, 1955). The output of the algorithm is the labeled tree matching obtained by removing all node alignments labeled with the special *none* relation.

6 Experiments

6.1 Experimental setup

Word alignment and full tree alignments are conceptually different tasks, which require partly different features and may have different practical applications. These are therefore addressed in separate experiments.

Table 2 summarizes the respective sizes of development and the held-out test set in terms of number of aligned graph pairs, number of aligned node pairs and number of tokens. The percentage of aligned nodes over all graphs is calculated relative to the number of nodes over all graphs. Since alignments involving non-terminal nodes are ignored in the task of word alignment, the number of aligned node pairs and the percentage of aligned nodes is lower in the word develop and word test sets. Table 3 gives the distribution of semantic relations in the development and test set, for word and tree alignment. It can be observed that the distribution if fairly skewed with Equals being the majority class, even more so for word alignments. Another thing to notice is that Intersects are much more frequent at the level of non-terminal alignments.

Data	Graph pairs	Node pairs	Tokens	Aligned nodes (%)
word develop	2 664	13 027	45 149	15.71
word test	547	2 858	10 005	14.96
tree develop	2 664	22 741	45 149	47.20
tree test	547	4 894	10 005	47.05

Table 2: Properties of develop and test data sets

Data	Eq	Re	Spec	Gen	Int
word develop	84.92	6.15	2.10	1.77	5.07
word test	85.62	6.09	2.17	1.99	4.13
tree develop	56.61	6.57	7.52	6.38	22.91
tree test	58.40	7.11	7.40	6.38	20.72

Table 3: Distribution of semantic similarity relations for word alignment and for full tree alignments in both develop and test data sets

Development was carried out using 10-fold cross validation on the development data and consequently reported scores on the development data are averages over 10 folds. Only two parameters were coarsely optimized on the development set. First, the amount of downsampling of the *none* class varied between 0.1 or 0.5. Second, the parameter k of the memory-based classifier – the number of nearest neighbours taken into account during classification – ranged from 1 to 15. Optimal settings were finally applied when testing on the held-out data.

A simple greedy alignment procedure served as baseline. For word alignment, identical words are aligned as Equals and identical roots as Restates. For full tree alignment, this is extended to the level of phrases so that phrases with identical words are aligned as Equals and phrases with identical roots as Restates. The baseline does not predict Spec-

ifies, Generalizes or Intersects relations, as that would require a more involved, knowledge-based approach.

All features used are described in Table 4. The word-based features rely on pure string processing and require no linguistic preprocessing. The morphology-based features exploit the limited amount of morphological analysis provided by the Alpino parser (Bouma et al., 2001). For instance, it provides word roots and decomposes compound words. Likewise the part-of-speech-based features use the coarse-grained part-of-speech tags assigned by the Alpino parser. The lexical-semantic features rely on the Cornetto database (Vossen et al., 2008), a recent extension to the Dutch WordNet, to look-up synonym and hypernym relations among source and target lemmas. Unfortunately there is no word sense disambiguation module to identify the correct senses. In addition, a background corpus of over 500M words of (mainly) news text provides the word counts required to calculate the Lin similarity measure (Lin, 1998). The syntax-based features use the syntactic structure, which is a mix of phrase-based and dependency-based analysis. The phrasal features express similarity between the terminal yields of source and target nodes. With the exception of *same-parent-lc-phrase*, these features are only used for full tree alignment, not for word alignment.

6.2 Results on word alignment

We evaluate our alignment model in two steps: first focussing on word alignment and then on full tree alignment. Table 5 summarizes the results for MBGM on word alignment (50% downsampling and $k = 3$), which we compare statistically to the baseline performance, and informally with the human scores reported in Table 1 in Section 4 (note that the human scores are only for a subset of the data used for automatic evaluation).

The first thing to observe is that the MBGM scores on the development and tests sets are very similar throughout. For predicting word alignments, the MBGM system performs significantly better than the baseline system ($t(18) = 17.72, p < .0001$). On the test set, MBGM obtains an F-score of nearly 89%, which is almost exactly halfway between the scores of the baseline system and the human scores. In a similar vein, the performance of the MBGM system on relation labeling is considerably better than that of the baseline system. For all semantic relations, MBGM performs significantly better than the baseline ($t(18) > 9.4138, p < .0001$ for each relation, trivially so for the Specifies, Generalizes and Intersects relations, which the baseline system never predicts).

The macro scores are plain averages over the 5 scores on each relation, whereas the micro scores are weighted averages. As the Equals is the majority class and at the same time easiest to predict, the micro scores are higher. The macro scores, however, better reflect performance on the real challenge, that is, correctly predicting the relations other than Equals. The MBGM macro average is 27.37% higher than the baseline (but still some 10% below the human top line), while the micro average is 5.83% higher and only 0.75% below the human top line. Macro scores on the test set are overall lower than those on the develop set, presumably because of tuning on the development data.

6.3 Results on tree alignment

Table 6 contains the results of full tree alignment (50% downsampling and $k = 5$); here both terminal and non-terminal nodes are aligned and classified in one pass. Again scores on the development and test set are very similar, the latter being slightly better. For full tree alignment, MBGM once again performs significantly better than the baseline, $t(18) = 25.68, p < .0001$. With an F-score on the test set of 86.65, MBGM scores almost 20 percent higher than the baseline system. This F-score is less than 2% lower than the average F-score obtained by our human annotators on full tree alignment, albeit not on exactly the same sample. The picture that emerges for semantic relation labeling is closely related to the one we saw for word alignments. MBGM significantly outperforms the baseline, for each semantic relation ($t(18) > 12.6636, p < .0001$). MBGM scores a macro average F-score of 52.24% (an increase of 30.05% over the baseline) and a micro average of 80.03% (12.68% above the base score). It is inter-

Feature	Type	Description
		Word
word-subsumption	string	indicate if source word equals, has as prefix, is a prefix of, has a suffix, is a suffix of, has as infix or is an infix of target word
shared-pre-/in-/suffix-len	int	length of shared prefix/infix/suffix in characters
source/target-stop-word	bool	test if source/target word is in a stop word list of frequent function words
source/target-word-len	int	length of source/target word in characters
word-len-diff	int	word length difference in characters
source/target-word-uniq	bool	test if source/target word is unique in source/target sentence
same-words-lhs/rhs	int	no. of identical preceding/following words in source and target word contexts
		Morphology
root-subsumption	string	indicate if source root equals, has as prefix, is a prefix of, has a suffix, is a suffix of, has as infix or is an infix of target root
roots-share-pre-/in-/suffix	bool	source and target root share a prefix/infix/suffix
		Part-of-speech
source/target-pos	string	source/target part-of-speech
same-pos	bool	test if source and target have same part-of-speech
source/target-content-word	bool	test if source/target word is a content word
both-content-word	bool	test if both source and target word are content words
		Lexical-semantic using Cornetto
cornet-restates	float	1.0 if source and target words are synonyms and 0.5 if they are near-synonyms, zero otherwise
cornet-specifies	float	Lin similarity score if source word is a hyponym of target word, zero otherwise
cornet-generalizes	float	Lin similarity score if source word is a hypernym of target word, zero otherwise
cornet-intersects	float	Lin similarity score if source word share a common hypernym, zero otherwise
		Syntax
source/target-cat	string	source/target syntactic category
same-cat	bool	test if source and target have same syntactic category
source/target-parent-cat	string	source/target syntactic category of parent node
source/target-deprel	string	source/target dependency relation
same-deprel	bool	test if source and target have same dependency relation
same-dephead-root	bool	test if the dependency heads of the source and target have same root
		Phrasal
word-prec/rec	float	precision/recall on the yields of source and target nodes
same-lc-phrase	bool	test if lower-cased yields of source and target nodes are identical
same-parent-lc-phrase	bool	test if lower-cased yields of parents of source and target nodes are identical
source/target-phrase-len	int	length of source/target phrase in words
phrase-len-diff	int	phrase length difference in words

Table 4: Features (where slashes indicate multiple versions of the same feature, e.g. *source/target-pos* represents the two features *source-pos* and *target-pos*)

esting to observe that MBGM obtains *higher* F-scores on Equals and on Intersects (the two most frequent relations) than the human annotators obtained. As a result of this, the micro F-score of the automatic full tree alignment is less than 2% lower than the human reference score.

Tree alignment can also be implemented as a two-step procedure, where in the first step alignments and semantic relation classifications at the word level are produced, while in the second step these are used to predict alignments and semantic relations for non-terminals. We experimented with such a two-step procedure as well, in one version using the actual word alignments and in the other the predicted word alignments. The scores of the two-step prediction are only marginally different from those of one step prediction, both for alignment and for relation classification, giving improvements in the order of about 1% for both subtasks. As is to be expected, the scores with true word alignments are much better than those with predicted word alignments. They are interesting though, because they suggest that a fairly good full tree alignment can be automatically ob-

		Alignment:	Labeling:						
			Eq:	Re:	Spec:	Gen:	Int:	Macro:	Micro:
Develop baseline:	Prec:	80.59	81.84	46.26	0.00	0.00	0.00	25.61	80.22
	Rec:	81.58	93.10	34.71	0.00	0.00	0.00	25.56	82.20
	F:	81.08	87.11	39.66	0.00	0.00	0.00	25.35	80.70
Develop MBGM:	Prec:	91.72	94.54	61.26	74.60	67.82	45.80	68.80	90.82
	Rec:	87.82	95.91	46.19	40.87	43.22	27.27	50.61	86.96
	F:	89.73	95.02	52.67	52.81	52.80	34.19	57.50	88.85
Test baseline:	Prec:	82.45	83.83	43.12	0.00	0.00	0.00	25.39	82.17
	Rec:	82.19	93.87	27.01	0.00	0.00	0.00	24.18	82.02
	F:	82.32	88.57	33.22	0.00	0.00	0.00	24.36	82.14
Test MBGM:	Prec:	90.92	94.20	53.33	59.87	54.21	42.47	60.84	89.90
	Rec:	87.09	95.41	40.21	32.75	43.28	20.31	46.39	86.11
	F:	88.96	94.80	45.85	42.34	48.17	27.48	51.73	87.97

Table 5: Scores (in percentages) on word alignment and semantic relation labeling

		Alignment:	Labeling:						
			Eq:	Re:	Spec:	Gen:	Int:	Macro:	Micro:
Develop baseline:	Prec:	82.50	83.76	46.72	0.00	0.00	0.00	26.10	82.18
	Rec:	54.54	93.66	20.01	0.00	0.00	0.00	22.74	54.34
	F:	65.67	88.43	28.02	0.00	0.00	0.00	23.29	65.42
Develop MBGM:	Prec:	92.23	96.15	55.90	54.40	56.15	70.33	66.59	84.99
	Rec:	81.04	94.03	26.64	21.71	29.34	70.27	48.40	74.68
	F:	86.27	95.08	36.08	31.03	38.54	70.30	54.21	79.50
Test baseline:	Prec:	84.23	85.68	42.24	0.00	0.00	0.00	25.58	84.14
	Rec:	56.21	94.44	14.08	0.00	0.00	0.00	21.70	56.15
	F:	67.43	89.85	21.12	0.00	0.00	0.00	22.19	67.35
Test MBGM:	Prec:	92.27	96.67	60.25	46.92	56.85	68.64	65.87	85.23
	Rec:	81.67	94.54	27.87	19.55	30.94	71.01	48.87	75.44
	F:	86.65	95.60	38.11	27.60	40.07	69.80	54.24	80.03

Table 6: Scores (in percentages) on full tree alignment and semantic relation labeling

tained given a manually checked word alignment.

7 Conclusions

We have proposed to analyse semantic similarity between comparable sentences by aligning their syntax trees, matching each node to the most similar node in the other tree (if any). In addition, alignments are labeled with a semantic similarity relation. We have presented a Memory-based Graph Matcher (MBGM) that performs both tasks simultaneously as a combination of exhaustive pairwise classification using a memory-based learning algorithm, and global optimization of alignments using a combinatorial optimization algorithm. It relies on a combination of morphological/syntactic analysis, lexical resources such as word nets, and machine learning using a parallel monolingual treebank. Results on aligning comparable news texts from a monolingual parallel treebank for Dutch show that MBGM consistently and significantly outperforms the baseline, both for alignment and labeling. This holds both for word alignment and tree alignment.

In future research we will test MBGM on other data, as the DAESO corpus contains sub-corpora with various degrees of semantic overlap. In addition, we intend to explore alternative features from word space models. Finally, we plan to evaluate MBGM in the context of NLP applications such as multi-document summarization. This includes work on how to define similarity at the sentence level in terms of the proportion of aligned constituents. Both MBGM and the annotated data set will be publicly released.[2]

Acknowledgments

This work was conducted within the DAESO project funded by the Stevin program (De Nederlandse Taalunie).

References

Barzilay, Regina and Kathleen R. McKeown. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328.

Bouma, Gosse, Gertjan van Noord, and Robert Malouf. 2001. Alpino: Wide-coverage computational analysis of Dutch. In Daelemans, Walter, Khalil Sima'an, Jorn Veenstra, and Jakub Zavre, editors, *Computational Linguistics in the Netherlands 2000.*, pages 45–59. Rodopi, Amsterdam, New York.

Daelemans, W., J. Zavrel, K. Van der Sloot, and A. Van den Bosch. 2009. TiMBL: Tilburg Memory Based Learner, version 6.2, reference manual. Technical Report ILK 09-01, Induction of Linguistic Knowledge, Tilburg University.

Dagan, I., O. Glickman, and B. Magnini. 2005. The PASCAL Recognising Textual Entailment Challenge. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*, Southampton, U.K.

Gildea, Daniel. 2003. Loosely tree-based alignment for machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 80–87, Sapporo, Japan.

Groves, D., M. Hearne, and A. Way. 2004. Robust subsentential alignment of phrase-structure trees. In *Proceedings of the 20th International Conference on Computational Linguistics (CoLing '04)*, pages 1072–1078.

Krahmer, Emiel, Erwin Marsi, and Paul van Pelt. 2008. Query-based sentence fusion is better defined and leads to more preferred results than generic sentence fusion. In Moore, J., S. Teufel, J. Allan, and S. Furui, editors, *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 193–196, Columbus, Ohio, USA.

Kuhn, Harold W. 1955. The Hungarian Method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97.

Lavie, A., A. Parlikar, and V. Ambati. 2008. Syntax-driven learning of sub-sentential translation equivalents and translation rules from parsed parallel corpora. In *Proceedings of the Second Workshop on Syntax and Structure in Statistical Translation*, pages 87–95.

Lin, D. 1998. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*, pages 296–304.

MacCartney, B. and C.D. Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 521–528.

MacCartney, Bill, Michel Galley, and Christopher D. Manning. 2008. A phrase-based alignment model for natural language inference. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 802–811, Honolulu, Hawaii, October.

Marsi, Erwin and Emiel Krahmer. 2005a. Classification of semantic relations by humans and machines. In *Proceedings of the ACL 2005 workshop on Empirical Modeling of Semantic Equivalence and Entailment*, pages 1–6, Ann Arbor, Michigan.

Marsi, Erwin and Emiel Krahmer. 2005b. Explorations in sentence fusion. In *Proceedings of the 10th European Workshop on Natural Language Generation*, Aberdeen, GB.

Radev, D.R. and K.R. McKeown. 1998. Generating natural language summaries from multiple on-line sources. *Computational Linguistics*, 24(3):469–500.

Tiedemann, J. and G. Kotzé. 2009. Building a Large Machine-Aligned Parallel Treebank. In *Eighth International Workshop on Treebanks and Linguistic Theories*, page 197.

Tinsley, J., V. Zhechev, M. Hearne, and A. Way. 2007. Robust language-pair independent sub-tree alignment. *Machine Translation Summit XI*, pages 467–474.

Vossen, P., I. Maks, R. Segers, and H. van der Vliet. 2008. Integrating lexical units, synsets and ontology in the Cornetto Database. In *Proceedings of LREC 2008*, Marrakech, Morocco.

Controlling Listening-oriented Dialogue using Partially Observable Markov Decision Processes

Toyomi Meguro[†], **Ryuichiro Higashinaka**[‡], **Yasuhiro Minami**[†], **Kohji Dohsaka**[†]
†NTT Communication Science Laboratories, NTT Corporation
‡NTT Cyber Space Laboratories, NTT Corporation
meguro@cslab.kecl.ntt.co.jp
higashinaka.ryuichiro@lab.ntt.co.jp
{minami,dohsaka}@cslab.kecl.ntt.co.jp

Abstract

This paper investigates how to automatically create a dialogue control component of a listening agent to reduce the current high cost of manually creating such components. We collected a large number of listening-oriented dialogues with their user satisfaction ratings and used them to create a dialogue control component using partially observable Markov decision processes (POMDPs), which can learn a policy to satisfy users by automatically finding a reasonable reward function. A comparison between our POMDP-based component and other similarly motivated systems using human subjects revealed that POMDPs can satisfactorily produce a dialogue control component that can achieve reasonable subjective assessment.

1 Introduction

Although task-oriented dialogue systems have been actively researched (Hirshman, 1989; Ferguson et al., 1996; Nakano et al., 1999; Walker et al., 2002), recently non-task-oriented functions are starting to attract attention, and systems without a specific task that deal with more casual dialogues, such as chats, are being actively investigated from their social and entertainment aspects (Bickmore and Cassell, 2001; Higashinaka et al., 2008; Higuchi et al., 2008).

In the same vein, we have been working on listening-oriented dialogues in which one conversational participant attentively listens to the other (hereafter, listening-oriented dialogue). Our aim is to build listening agents that can implement such a listening process so that users can satisfy their desire to speak and be heard. Figure 1 shows an excerpt from a typical listening-oriented dialogue. In the literature, dialogue control components for less (or non-) task-oriented dialogue systems, such as listening agents, have typically used hand-crafted rules for dialogue control, which can be problematic because completely covering all dialogue states by hand-crafted rules is difficult when the dialogue has fewer task restrictions (Wallace, 2004; Isomura et al., 2009).

To solve this problem, this paper aims to automatically build a dialogue control component of a listening agent using partially observable Markov decision processes (POMDPs). POMDPs, which make it possible to learn a policy that can maximize the averaged reward in partially observable environments (Pineau et al., 2003), have been successfully adopted in task-oriented dialogue systems for learning a dialogue control module from data (Williams and Young, 2007). However, no work has attempted to use POMDPs for less (or non-) task-oriented dialogue systems, such as listening agents, because user goals are not as well-defined as task-oriented ones, complicating the finding of a reasonable reward function.

We apply POMDPs to listening-oriented dialogues by having the system learn a policy that simultaneously maximizes how well users feel that they are being listened to (hereafter, user satisfaction) and how smoothly the system generates dialogues (hereafter, smoothness). This formulation is new; no work has considered both user satisfaction and smoothness using POMDPs. We collected a large amount of listening-oriented dialogues and annotated them with dialogue acts and also obtained subjective evaluation results for them. From them, we calculated the rewards and learned the POMDP policies. We evaluated the dialogue-act tag sequences of our POMDPs using human subjects.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 761–769,
Beijing, August 2010

	Utterance	Dialogue act
S:	Good evening.	GREETING
	The topic is "food," nice to meet you.	GREETING
L:	Nice to meet you, too.	GREETING
S:	I had curry for dinner.	S-DISC (sub: fact)
	Do you like curry?	QUESTION (sub: pref)
L:	Yes, I do.	SYMPATHY
S:	Really?	REPEAT
	Me, too.	SYMPATHY
L:	Do you usually go out to eat?	QUESTION (sub: habit)
S:	No, I always cook at home.	S-DISC (sub: habit)
	I don't use any special spices, but I sometimes cook noodles using soup and curry.	S-DISC (sub: habit)
L:	That sounds good!	S-DISC (sub: pref (positive))

Figure 1: Excerpt of a typical listening-oriented dialogue. Dialogue topic is "food." Dialogue acts corresponding to utterances are shown in parentheses (See Table 1 for meanings): S-DISC stands for SELF-DISCLOSURE; PREF for PREFERENCE; S for speaker; and L for listener. The dialogue was translated from Japanese by the authors.

The next section introduces related work. Section 3 describes our approach. Section 4 describes our collection of listening-oriented dialogues. This is followed in Section 5 by an evaluation experiment that compared our POMDP-based dialogue control with other similarly motivated systems. The last section summarizes the main points and mentions future work.

2 Related work

With increased attention on social dialogues and senior peer counseling, work continues to emerge on listening-oriented dialogues. One early work is (Maatman et al., 2005), which showed that virtual agents can give users the sense of being heard using such gestures as nodding and head shaking. Recently, Meguro et al. (2009a) analyzed the characteristics of listening-oriented dialogues. They compared listening-oriented dialogues and casual conversations between humans, revealing that the two types of dialogues have significantly different flows and that listeners actively question with frequently inserted self-disclosures; the speaker utterances were mostly concerned with self-disclosure.

Shitaoka et al. (2010) also investigated the functions of listening agents, focusing on their response generation components. Their system takes the confidence score of speech recognition into account and changes the system response accordingly; it repeats the user utterance or makes an empathic utterance for high-confidence user utterances and makes a backchannel when the confidence is low. The system's empathic utterances can be "I'm happy" or "That's too bad," depending on whether a positive or negative expression is included in the user utterances. Their system's response generation only uses the speech recognition confidence and the polarity of user utterances as cues to choose its actions. Currently, it does not consider the utterance content or the user intention.

In order for listening agents to achieve high smoothness, a switching mechanism between the "active listening mode," in which the system is a listener, and the "topic presenting mode," in which the system is a speaker, has been proposed (Yokoyama et al., 2010; Kobayashi et al., 2010). Here, the system uses a heuristic function to maintain a high user interest level and to keep the system in an active listening mode. Dialogue control is done by hand-crafted rules. Our motivation bears some similarity to theirs in that we want to build a listening agent that gives users a sense of being heard; however, we want to automatically make such an agent from dialogue data.

POMDPs have been introduced for robot action control (Pineau et al., 2003). Here, the system learns to make suitable movements for completing a certain task. Over the years, POMDPs have been actively studied for applications to spoken dialogue systems. Williams et al. (2007) successfully used a POMDP for dialogue control in a ticket-buying domain in which the objective was to fix the departure and arrival places for tickets. Recent work on POMDPs indicates that it is possible to train a dialogue control module in task-oriented dialogues when the user goal is obvious. In contrast, in this paper, we aim to verify whether POMDPs can be applied to less task-oriented dialogues (i.e., listening-oriented dialogues) where user goals are not as obvious.

In a recent study, Minami et al. (2009) applied POMDPs to non-task-oriented man-machine interaction. Their system learned suitable action control of agents that can act smoothly by obtaining rewards from the statistics of artificially generated data. Our work is different because we use real human-human dialogue data to

train POMDPs for dialogue control in listening-oriented dialogues.

3 Approach

A typical dialogue system has utterance understanding, dialogue control, and utterance generation modules. The utterance understanding module comprehends user natural-language utterances, whose output (i.e., a user dialogue act) is passed to the dialogue control module. The dialogue control module chooses the best system dialogue act at every dialogue point using the user dialogue act as input. The utterance generation module generates natural-language utterances and says them to users by realizing the system dialogue acts as surface forms.

This paper focuses on the dialogue control module of a listening agent. Since a listening-oriented dialogue has a characteristic conversation flow (Meguro et al., 2009a), focusing on this module is crucial because it deals with the dialogue flow. Our objective is to train from data a dialogue control module that achieves a smooth dialogue flow that makes users feel that they are being listened to attentively.

3.1 Dialogue control using POMDPs

The purpose of our dialogue control is to simultaneously create situations in which users feel listened to (i.e., user satisfaction) and to generate smooth action sequences (i.e., smoothness). To do this, we automatically and statistically train the reward and the policy of the POMDP using a large amount of listening-oriented dialogue data. POMDP is a reinforcement learning framework that can learn a policy to select an action sequence that maximizes average future rewards. Setting a reward is crucial in POMDPs.

For our purpose, we introduce two different rewards: one for user satisfaction and the other for smoothness. Before creating a POMDP structure, we used the dynamic Bayesian network (DBN) structure (Fig. 2) to obtain the statistical structure of the data and the two rewards.

The random values in the DBN are as follows: s_o and s_a are the dialogue state and action state, o is a speaker observation, a is a listener action, and d is a random variable for an evaluation score that indicates the degree of the user being listened to. This evaluation score can be obtained by questionnaires, and the variable is used for calculating a user satisfaction reward for the POMDP. The DBN arcs in Fig. 2 define the emission and transition probabilities. $\Pr(o'|s'_o)$ is the emission probability of o' given s'_o. $\Pr(d|s_o)$ is the emission probability of d given s_o. $\Pr(s'_o|s_o, a)$ is a transition probability from s_o to s'_o given a. The DBN is trained using the EM algorithm. Using the obtained variables, we calculate the two reward functions as follows:

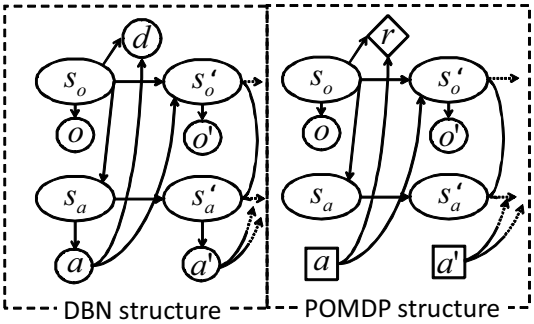


Figure 2: DBN and POMDP structures employed in this paper. Note that a in the POMDP is isolated from other states because it is decided by a learned policy.

(1) Reward for user satisfaction This reward is obtained from the d variable by

$$r_1((s_o, *), a) = \sum_{d=min}^{max} d \times \Pr(d|s_o, a),$$

where * is arbitrary s_a and min and max are minimum and maximum evaluation scores.

(2) Reward for smoothness For smoothness, we maximize the action predictive probability given the history of actions and observations. The probability is calculated from listening-oriented dialogue data. s_a is introduced for estimating the predictive probability of action a and for selecting a to maximize the predictive probability.

We set $\Pr(a|s_a) = 1$ when $a = s_a$ so that s_a corresponds one-on-one with a. Then, if $a_t = s_a$ at time t is given, we obtain

$$\begin{aligned}
&\Pr(a_t|o_1, a_1, \ldots, a_{t-1}, o_t) \\
&\quad = \sum_{s'_a} \Pr(a_t|s'_a) \Pr(s'_a|o_1, a_1, \ldots, a_{t-1}, o_t) \\
&\quad = \Pr(s_a|o_1, a_1, \ldots, o_{t-1}, a_{t-1}, o_t)
\end{aligned}$$

Consequently, maximizing the predictive probability of a equals maximizing that of s_a. If we

set 1.0 to reward $r_2((*, s_a), a)$ when $s_a = a$, the POMDP will generate actions that maximize their predictive probabilities. We believe that this reward should increase the smoothness of a system action sequence since the sequence is generated according to the statistics of human-human dialogues.

Converting a DBN into a POMDP The DBN is converted into a POMDP (Fig. 2), while maintaining the transition and output probabilities. We convert d to r as described above.

The system is in a partially observed state. Since the state is not known exactly, we use a distribution called "belief state" b_t with which we obtain the average reward that will be gained in the future at time t by:

$$V_t = \sum_{\tau=0}^{\infty} \gamma^{\tau} \sum_{s} b_{\tau+t}((s_o, s_a)) r((s_o, s_a), a_{\tau+t}),$$

where τ is a discount factor; namely, the future reward is decreased by τ. A policy is learned by value iteration so that the action that maximizes V_t can be chosen. We define $r((s_o, s_a), a)$ as follows:

$$r((s_o, s_a), a) = r_1((s_o, *), a) + r_2((*, s_a), a).$$

By balancing these two rewards, we can choose an action that satisfies both user satisfaction and smoothness.

4 Data collection

We collected listening-oriented dialogues using human subjects who consisted of ten listeners (five males and five females) and 37 speakers (18 males and 19 females). The listeners and speakers ranged from 20 to 60 years old and were all native Japanese speakers. Listeners and speakers were matched to form a listener-speaker pair and communicated over the Internet with our chat interface. They used only text; they were not allowed to use voice, video, or facial expressions. The speakers chose their own listener and freely participated in dialogues from 7:00 pm to midnight for a period of 15 days. One conversation was restricted to about ten minutes. The subjects talked about a topic chosen by the speaker. There were 20 predefined topics: money, sports, TV and radio, news, fashion, pets, movies, music, housework and childcare, family, health, work, hobbies, food, human relationships, reading, shopping, beauty aids, travel, and miscellaneous. The listeners were instructed to make it easy for the speakers to say what the speakers wanted to say. We collected 1260 listening-oriented dialogues.

4.1 Dialogue-act annotation

We labeled the collected dialogues using the dialogue-act tag set shown in Table 1. We made these tags by selecting, extending, and modifying those from previous studies that concerned human listening behaviors in some way (Meguro et al., 2009a; Jurafsky et al., 1997; Ivey and Ivey, 2002). In our tag set, only question and self-disclosure tags have sub-category tags. Two annotators (not the authors) labeled each sentence of our collected dialogues using these 32 tags. In dialogue-act annotation, since there can be several sentences in one utterance, one annotator first split the utterances into sentences, and then both annotators labeled each sentence with a single dialogue act.

4.2 Obtaining evaluation scores

POMDPs need evaluation scores (i.e., d) for dialogue acts (i.e., a) for training a reward function. Therefore, we asked a third-party participant, who was neither a listener nor a speaker in our dialogue data collection, to evaluate the user satisfaction levels of the collected dialogues. She rated each dialogue in terms of how she would have felt "being heard" after the dialogue if she had been the speaker of the dialogue in question. She provided ratings on the 7-point Likert scale for each dialogue. Since she rated the whole dialogue with a single rating, we set the evaluation score of each action within a dialogue using the evaluation score for that dialogue.

We used a third-person's evaluation and not the original person's to avoid the fact that the evaluative criterion is too different between humans; identical evaluation scores from two people do not necessarily reflect identical user satisfaction levels. We highly valued the reliability and consistency of the third-person scores. This way, at least, we can train a policy that maximizes its average reward function for the rater, which we need to verify first before considering adaptation to two or more individuals.

5 Experiment

5.1 Experimental setup

The experiment followed three steps.

GREETING	Greeting and confirmation of dialogue theme. e.g., Hello. Let's talk about lunch.
INFORMATION	Delivery of objective information. e.g., My friend recommended a restaurant.
SELF-DISCLOSURE	Disclosure of preferences and feelings.
sub: fact	e.g., I live in Tokyo.
sub: experience	e.g., I had a hamburger for lunch.
sub: habit	e.g., I always go out for dinner.
sub: preference (positive)	e.g., I like hamburgers.
sub: preference (negative)	e.g., I don't really like hamburgers.
sub: preference (neutral)	e.g., Its taste is near my homemade taste.
sub: desire	e.g., I want to try it.
sub: plan	e.g., I'm going there next week.
sub: other	
ACKNOWLEDGMENT	Encourage the conversational partner to speak. e.g., Well. Aha.
QUESTION	Utterances that expect answers.
sub: information	e.g., Please tell me how to cook it.
sub: fact	e.g., What kind of curry?
sub: experience	e.g., What did you have for dinner?
sub: habit	e.g., Did you cook it yourself?
sub: preference	e.g., Do you like it?
sub: desire	e.g., Don't you want to eat rice?
sub: plan	e.g., What are you going to have for dinner?
sub: other	
SYMPATHY	Sympathetic utterances and praises. e.g., Me, too.
NON-SYMPATHY	Negative utterances. e.g., Not really.
CONFIRMATION	Confirm what the conversation partner said. e.g., Really?
PROPOSAL	Encourage the partner to act. e.g., Try it.
REPEAT	Repeat the partner's utterance.
PARAPHRASE	Paraphrase the partner's utterance.
APPROVAL	Broach or show goodwill toward the partner. e.g., Absolutely!
THANKS	Express thanks e.g., Thank you.
APOLOGY	Express regret e.g., I'm sorry.
FILLER	Filler between utterances. e.g., Uh. Let me see.
ADMIRATION	Express affection. e.g., Ha-ha.
OTHER	Other utterances.

Table 1: Definition and example of dialogue acts

In the first step, we created our POMDP system using our approach (See Section 3.1). We also made five other systems for comparison that we describe in Section 5.2. Each system outputs dialogue-act tag sequences for evaluation. The dialogue theme was "food" because it was the most frequent theme and accounted for 20% of our data (See Table 2 for the statistics); we trained our POMDP using the "food" dialogues. We restricted the dialogue topic to verify that our approach at least works with a small set. Since there is no established measure for automatically evaluating a dialogue-act tag sequence, we evaluated

	All	Food (subset of All)
# dialogues	1260	250
# words	479881	94867
# utterances per dialogue	28.2	29.1
# dialogues per listener	126	25
# dialogues per speaker	34	6.8
# dialogue acts	67801	13376
inter-annotator agreement	0.57	0.55

Table 2: Statistics of collected dialogues and dialogue-act annotation. Inter-annotator agreement means agreement of dialogue-act annotation using Cohen's κ.

our dialogue control module using human subjective evaluations. However, this is very difficult to do because dialogue control modules only output dialogue acts, not natural language utterances.

In the second step, we recruited participants who created natural language utterances from dialogue-act tag sequences. In their creating dialogues, we provided them with situations to stimulate their imaginations. Table 3 shows the situations, which were deemed common in everyday Japanese life; we let the participants create utterances that fit the situations. These situations were necessary because, without restrictions, the evaluation scores could be influenced by dialogue content rather than by dialogue flow.

For this dialogue-imagining exercise, we recruited 16 participants (eight males and eight females) who ranged from 19 to 39 years old. Each participant made twelve dialogues using two situations. For assigning the situations, we first created four conditions: (1) a student and living with family, (2) working and living with family, (3) a student and living alone, and (4) working and living alone. Then the participants were categorized into one of these conditions on the basis of their actual lifestyle and assigned two of the situations matching the condition.

For each situation, each participant created six imaginary dialogues from the six dialogue-act sequences output by the six systems: our POMDP and the other five systems for comparison. This process produced such dialogues as shown in Figs. 5 and 6. The dialogue in Fig. 5 was made from a dialogue-act tag sequence of a human-human conversation using No. 1 of Table 3. The dialogue in Fig. 6 was made from the sequence of our POMDP using No. 2 of Table 3.

In the third step, we additionally recruited three judges (one male and two females) to evalu-

ate the imagined 192 ($16 \times 2 \times 6$) dialogues. The judges were neither the participants who made dialogues nor those who rated the collected listening-oriented dialogues. Six dialogues made from one situation were randomly shown to the judges one-by-one, who then filled out questionnaires to indicate their user satisfaction levels by answering this question on a 7-point Likert scale: "If you had been the speaker, would you have felt that you were listened to?"

5.2 Systems for comparison

We created our POMDP-based dialogue control and five other systems for comparison.

POMDP We learned a policy based on our approach. We used "food" dialogues (See Section 4), and the evaluation scores were those described in Section 4.2. This system used the policy to generate sequences of dialogue-act tags by simulation; user observations were generated based on emission probability, and system actions were generated based on the policy.

In this paper, the total number of observations and actions was 33 because we have 32 dialogue-act tags (See Table 1) plus a "skip" tag. In learning the policy, an observation and an action must individually take turns, but our data can include multiple dialogue-act tags in one utterance. Therefore, if there is more than one dialogue-act tag in one utterance, a "skip" is inserted between the tags. The state numbers for S_o and S_a were 16 and 33, respectively. In this experiment, we set 10 to $r_2((*, s_a), a)$.

EvenPOMDP We arranged a POMDP using only the smoothness reward (hereafter, EvenPOMDP) by creating a POMDP system with a fixed evaluation score; hence user satisfaction is not incorporated in the reward. When using fixed (even) evaluation scores for all dialogues, the effect of the user satisfaction reward is denied, and the system only generates highly frequent sequences. We have EvenPOMDP to clarify whether user satisfaction is necessary. The other conditions are identical as in the POMDP system.

HMM We modeled our dialogue-act tag sequences using a Speaker HMM (SHMM) (Meguro et al., 2009a), which has been utilized to model two-party listening-oriented dialogues. In a SHMM, half the states emit listener dialogue acts, and the other half emit speaker dialogue acts. All states are connected to each other. We modeled the "food" dialogues using an SHMM, and made the model generate the most probable dialogue-act tag sequences. More specifically, first, a dialogue-act tag was generated randomly based on the initial state. If the state was that of a listener, we generated a maximum likelihood action and the state was randomly transited based on the transition probability. If the state was that of a speaker, we randomly generated an action based on the emission probability and the state was transited using the maximum likelihood transition probability.

Figure 3: Structure of rule-based system

Rule-based system This system creates dialogue-act tag sequences using hand-crafted rules that are based on the findings in (Meguro et al., 2009a) and are realized as shown in Fig. 3. A sequence begins at state ① in Fig. 3, and one dialogue act is generated at each state. At state ③, a sub-category tag under QUESTION is chosen randomly, and at state ④, a matched sub-category tag under SELF-DISCLOSURE is chosen. At state ⑤, the listener's SELF-DISCLOSURE or SYMPATHY is generated randomly.

Human dialogue sequence This system created dialogue-act tag sequences by randomly choosing dialogues between humans from the collected data and used their annotated tag sequences.

Random This system simply created dialogue-act tag sequences at random.

5.3 Experimental results

Figure 4 shows the average subjective evaluation scores. Except between HMM and EvenPOMDP, there was a significant difference ($p<0.01$) between all systems in a non-parametric multiple comparison test (Steel-Dwass test). The dialogues shown in Figs. 5 and 6 were generated by the systems. The dialogue in Fig. 5 was made from human dialogue sequences, and the one in Fig. 6 was made from POMDP.

	With whom	What day	What time	What	Where	Who made
1	family	weekday	around 6:00 pm	grilled salmon	home	mother
2	family	weekend	around 7:00 pm	potato and meat	home	mother
3	co-workers	weekday	at noon	boiled seaweed	lunch box	myself
⋮	⋮	⋮	⋮	⋮	⋮	⋮
32	friend	weekday	at noon	hamburger	school cafeteria	N/A

Table 3: Dialogue situations relating to everyday Japanese life

We qualitatively analyzed the dialogues of each system and observed the following characteristics:

POMDP At a dialogue's beginning, the system greets several times and shifts to a different phase in which listeners ask questions and self-disclose to encourage speakers to reciprocate.

Rule-based The output of this system seems very natural and easy to read. The dialogue-act tags followed reasonable rules, making it easier for the participants to create natural utterances from them.

Human conversation The dialogues between humans were obviously natural before they were changed to tags from the natural-language utterances. However, human dialogues have randomness, which makes it difficult for the participants to create natural-language utterances from the tags. Hence, the evaluation score for this system was lower than the "Rule-based."

HMM, EvenPOMDP Since these systems continually output the same action tags, their output was very unnatural. For example, greetings never stopped because GREETING is most frequently followed by GREETING in the data. These systems have no mechanism to stop this loop.

POMDP successfully avoided such continuation because its actions have more varied rewards. For example, GREETING is repeated in EvenPOMDP because its smoothness reward is high; however, in POMDP, although the smoothness reward remains high, its user satisfaction reward is not that high. This is because greetings appear in all dialogues and their user satisfaction reward converges to the average. Therefore, such actions as greetings do not get repeated in POMDP. In POMDP, some states have high user satisfaction rewards, and the POMDP policy generated actions to move to such states.

Random Since this system has more variety of tags than HMM, its evaluation scores outperformed HMM, but were outperformed by POMDP, which learned statistically from the data.

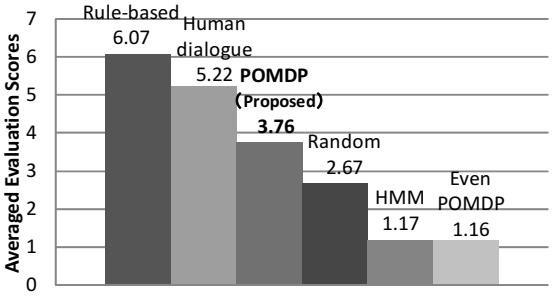


Figure 4: System scores. Except between POMDP and EvenPOMDP, significant differences exist among all systems ($p<0.01$).

From our qualitative analysis, we found that POMDP can generate more satisfying sequences than HMM/EvenPOMDP because it does not fall into the loop of frequent dialogue-act tag sequences. This suggests the usefulness of incorporating two kinds of rewards into the policy and that our approach for setting a reward is promising.

However, with the proposed POMDP, unnatural sequences remain; for example, the system suddenly output THANKS, as shown in Fig. 6. The number of states may have been too small. We plan to investigate what caused this in the future.

In our qualitative analysis, we observed that randomness in dialogues might hold a clue for improving evaluation scores. Therefore, we measured the perplexity of each system output using dialogue-act trigrams and obtained 72.8 for "Random," 27.4 for "Human dialogue," 7.4 for "POMDP," 3.2 for "HMM," 2.5 for "EvenPOMDP," and 1.7 for "Rule-based."

The perplexity of the human dialogues is less than that of the random system, but humans also exhibit a certain degree of freedom. On the other hand, POMDP's perplexity is less than the human dialogues; they still have some freedom, which probably led to their reasonable evaluation scores. Considering that HMM and EvenPOMDP, which continually output the same dialogue acts, had low

	Utterance	Dialogue act
S:	Hello.	GREETING
L:	Nice to meet you	GREETING
S:	I had dinner at home today.	S-DISC (sub: fact)
	Do you like grilled salmon?	QUESTION, PREF
L:	Yes, I think so.	SYMPATHY
	I sometimes want to have a fancy meal.	S-DISC (sub: desire)
S:	Deluxe.	REPEAT
	Me too.	SYMPATHY
L:	Do you usually do your own cooking?	QUESTION (sub: habit)
S:	No, I don't.	S-DISC, HABIT
	I always buy my meals at the convenience store.	S-DISC (sub: habit)
L:	I like the lunch boxes of convenience stores	S-DISC (sub: pref (positive))

Figure 5: Excerpt of listening-oriented dialogue that participant imagined from tag sequences of human conversations. Dialogue was translated from Japanese by the authors.

	Utterance	Dialogue act
L:	Nice to meet you.	GREETING
	Where and who did you have dinner with today?	QUESTION (sub: fact)
S:	I had "niku-jaga" (meat and beef) with my family at home.	S-DISC (sub: fact)
L:	Oh.	ADMIRATION
S:	I think it is normal to eat with your family at home.	S-DISC (sub: pref (neutral))
L:	Thanks.	THANKS
	Do you have any brothers or sisters?	QUESTION (sub: fact)
	Soon, my brother and his wife will visit my home.	S-DISC (sub: plan)
S:	I see.	SYMPATHY
L:	I want to use expensive meat in my "niku-jaga."	S-DISC (sub: desire)
	Oh.	ADMIRATION
	Please give me your recipe.	QUESTION (sub: information)
S:	My friends claim that my "niku-jaga" is as good as a restaurant's.	INFORMATION
L:	I'd love to try it	S-DISC (sub: desire)

Figure 6: Excerpt of a listening-oriented dialogue made from tag sequences of POMDP

evaluation scores, we conclude that randomness is necessary in non-task-oriented dialogues and that some randomness can be included with our approach. We do not discuss "Rule-based" here because its tag sequence was meant to have small perplexity.

6 Conclusion and Future work

This paper investigated the possibility of automatically building a dialogue control module from dialogue data to create automated listening agents.

With a POMDP as a learning framework, a dialogue control module was learned from the listening-oriented dialogues we collected and compared with five different systems. Our POMDP system showed higher performance in subjective evaluations than other statistically motivated systems, such as an HMM-based one, that work by selecting the most likely subsequent action in the dialogue data. When we investigated the output sequences of our POMDP system, the system frequently chose to self-disclose and question, which corresponds to human listener behavior, as revealed in the literature (Meguro et al., 2009a). This suggests that learning dialogue control by POMDPs is achievable for listening-oriented dialogues.

The main contribution of this paper is that we successfully showed that POMDPs can be used to train dialogue control policies for less task-oriented dialogue systems, such as listening agents, where the user goals are not as clear as task-oriented ones. We also revealed that the reward function can be learned effectively by our formulation that simultaneously maximizes user satisfaction and smoothness. Finding an appropriate reward function is a real challenge for less task-oriented dialogue systems; this work has presented the first workable solution.

Much work still remains. Even though we conducted an evaluation experiment by simulation (i.e, offline evaluation), human dialogues obviously do not necessarily proceed as in simulations. Therefore, we plan to evaluate our system using online evaluation, which also forces us to implement utterance understanding and generation modules. We also want to incorporate the idea of topic shift into our policy learning because we observed in our data that listeners frequently change topics to keep speakers motivated. We are also considering adapting the system behavior to users. Specifically, we want to investigate dialogue control that adapts to the personality traits of users because it has been found that the flow of listening-oriented dialogues differs depending on the personality traits of users (Meguro et al., 2009b). Finally, although we only dealt with text, we also want to extend our approach to speech and other modalities, such as gestures and facial expressions.

References

Bickmore, Timothy and Justine Cassell. 2001. Relational agents: a model and implementation of building user trust. In *Proc. SIGCHI conference on human factors in computing systems (CHI)*, pages 396–403.

Ferguson, George, James F. Allen, and Brad Miller. 1996. TRAINS-95: towards a mixed-initiative planning assistant. In *Proc. Third Artificial Intelligence Planning Systems Conference (AIPS)*, pages 70–77.

Higashinaka, Ryuichiro, Kohji Dohsaka, and Hideki Isozaki. 2008. Effects of self-disclosure and empathy in human-computer dialogue. In *Proc. IEEE Workshop on Spoken Language Technology (SLT)*, pages 108–112.

Higuchi, Shinsuke, Rafal Rzepka, and Kenji Araki. 2008. A casual conversation system using modality and word associations retrieved from the web. In *Proc. 2008 conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 382–390.

Hirshman, Lynette. 1989. Overview of the DARPA speech and natural language workshop. In *Proc. DARPA Speech and Natural Language Workshop 1989*, pages 1–2.

Isomura, Naoki, Fujio Toriumi, and Kenichiro Ishii. 2009. Evaluation method of non-task-oriented dialogue system using HMM. *IEICE Transactions on Information and Systems*, J92-D(4):542–551.

Ivey, Allen E. and Mary Bradford Ivey. 2002. *Intentional Interviewing and Counseling: Facilitating Client Development in a Multicultural Society*. Brooks/Cole Publishing Company.

Jurafsky, Dan, Elizabeth Shriberg, and Debra Biasca, 1997. *Switchboard SWBD-DAMSL Shallow-Discourse-Function Annotation Coders Manual*.

Kobayashi, Yuka, Daisuke Yamamoto, Toshiyuki Koga, Sachie Yokoyama, and Miwako Doi. 2010. Design targeting voice interface robot capable of active listening. In *Proc. 5th ACM/IEEE international conference on Human-robot interaction (HRI)*, pages 161–162,

Maatman, R. M., Jonathan Gratch, and Stacy Marsella. 2005. Natural behavior of a listening agent. *Lecture Notes in Computer Science*, 3661:25–36.

Meguro, Toyomi, Ryuichiro Higashinaka, Kohji Dohsaka, Yasuhiro Minami, and Hideki Isozaki. 2009a. Analysis of listening-oriented dialogue for building listening agents. In *Proc. 10th Annual SIGDIAL Meeting on Discourse and Dialogue (SIGDIAL)*, pages 124–127.

Meguro, Toyomi, Ryuichiro Higashinaka, Kohji Dohsaka, Yasuhiro Minami, and Hideki Isozaki. 2009b. Effects of personality traits on listening-oriented dialogue. In *Proc. International Workshop on Spoken Dialogue Systems Technology (IWSDS)*, pages 104–107.

Minami, Yasuhiro, Akira Mori, Toyomi Meguro, Ryuichiro Higashinaka, Kohji Dohsaka, and Eisaku Maeda. 2009. Dialogue control algorithm for ambient intelligence based on partially observable markov decision processes. In *Proc. International Workshop on Spoken Dialogue Systems Technology (IWSDS)*, pages 254–263.

Nakano, Mikio, Noboru Miyazaki, Jun ichi Hirasawa, Kohji Dohsaka, and Takeshi Kawabata. 1999. Understanding unsegmented user utterances in real-time spoken dialogue systems. In *Proc. 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 200–207.

Pineau, Joelle., Geoff. Gordon, and Sebastian Thrun. 2003. Point-based value iteration: An anytime algorithm for POMDPs. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1025–1032.

Shitaoka, Kazuya, Ryoko Tokuhisa, Takayoshi Yoshimura, Hiroyuki Hoshino, and Narimasa Watanabe. 2010. Active listening system for dialogue robot. In *JSAI SIG-SLUD Technical Report*, volume 58, pages 61–66. (in Japanese).

Walker, Marilyn, Alex Rudnicky, John Aberdeen, Elizabeth Owen Bratt, Rashmi Prasad, Salim Roukos, Greg S, and Seneff Dave Stallard. 2002. DARPA communicator evaluation: progress from 2000 to 2001. In *Proc. International Conference on Spoken Language Processing (ICSLP)*, pages 273–276.

Wallace, Richard S. 2004. *The Anatomy of A.L.I.C.E.* A.L.I.C.E. Artificial Intelligence Foundation, Inc.

Williams, Jason D. and Steve Young. 2007. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422.

Yokoyama, Sachie, Daisuke Yamamoto, Yuka Kobayashi, and Miwako Doi. 2010. Development of dialogue interface for elderly people –switching the topic presenting mode and the attentive listening mode to keep chatting–. In *IPSJ SIG Technical Report*, volume 2010-SLP-80, pages 1–6. (in Japanese).

Recognising Entailment within Discourse

Shachar Mirkin[§], **Jonathan Berant**[†], **Ido Dagan**[§], **Eyal Shnarch**[§]

§ Computer Science Department, Bar-Ilan University
† The Blavatnik School of Computer Science, Tel-Aviv University

Abstract

Texts are commonly interpreted based on the entire discourse in which they are situated. Discourse processing has been shown useful for inference-based application; yet, most systems for textual entailment – a generic paradigm for applied inference – have only addressed discourse considerations via off-the-shelf coreference resolvers. In this paper we explore various discourse aspects in entailment inference, suggest initial solutions for them and investigate their impact on entailment performance. Our experiments suggest that discourse provides useful information, which significantly improves entailment inference, and should be better addressed by future entailment systems.

1 Introduction

This paper investigates the problem of recognising textual entailment within discourse. Textual Entailment (TE) is a generic framework for applied semantic inference (Dagan et al., 2009). Under TE, the relationship between a *text* (T) and a textual assertion (*hypothesis*, H) is defined such that *T entails H* if humans reading T would infer that H is most likely true (Dagan et al., 2006).

TE has been successfully applied to a variety of natural language processing applications, including information extraction (Romano et al., 2006) and question answering (Harabagiu and Hickl, 2006). Yet, most entailment systems have thus far paid little attention to discourse aspects of inference. In part, this is the result of the unavailability of adept tools for handling the kind of discourse processing required for inference. In addition in the main TE benchmarks, the Recognising Textual Entailment (RTE) challenges, discourse played little role. This state of affairs has started to change with the recent introduction of the RTE Pilot "Search" task (Bentivogli et al., 2009b), in which assessed texts are situated within complete documents. In this setting, texts need to be interpreted based on their entire discourse (Bentivogli et al., 2009a), hence attending to discourse issues becomes essential. Consider the following example from the task's dataset:

(T) *The seven men on board were said to have as little as 24 hours of air.*

For the interpretation of T, e.g. the identity and whereabouts of the seven men, one must consider T's discourse. The preceding sentence T', for instance, provides useful information to that aim:

(T') *The Russian navy worked desperately to save a small military submarine*.

This example demonstrates a common situation in texts, and is also applicable to the RTE Search task's setting. Still, little was done by the task's participants to consider discourse, and sentences were mostly processed independently.

Analyzing the Search task's development set, we identified several key discourse aspects that affect entailment in a discourse-dependent setting. First, we observed that the coverage of available coreference resolution tools is considerably limited. To partly address this problem, we extend the set of coreference relations to phrase pairs with a certain degree of lexical overlap, as long as no semantic incompatibility is found between them. Second, many bridging relations (Clark, 1975) are realized in the form of "global information" perceived as known for entire documents. As bridging falls completely out of the scope of available resolvers, we address this phenomenon by identifying and weighting prominent document terms and allowing their incorporation in inference even

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 770–778,
Beijing, August 2010

when they are not explicitly mentioned in a sentence. Finally, we observed a coherence-related discourse phenomenon, namely inter-relations between entailing sentences in the discourse, such as the tendency of entailing sentences to be adjacent to one another. To that end, we apply a two-phase classification scheme, where a second-phase meta-classifier is applied, extracting discourse and document-level features based on the classification of each sentence on its own.

Our results show that, even when simple solutions are employed, the reliance on discourse-based information is helpful and achieves a significant improvement of results. We analyze the contribution of each component and suggest some future work to better attend to discourse in entailment systems. To our knowledge, this is the most extensive effort thus far to empirically explore the effect of discourse on entailment systems.

2 Background

Discourse plays a key role in text understanding applications such as question answering or information extraction. Yet, such applications typically only handle a narrow aspect of discourse, addressing coreference by term substitution (Dali et al., 2009; Li et al., 2009). The limited coverage and scope of existing tools for coreference resolution and the unavailability of tools for addressing other discourse aspects also contribute to this situation. For instance, VP anaphora and bridging relations are usually not handled at all by such resolvers. A similar situation is seen in the TE research field.

The prominent benchmark for entailment systems evaluation is the series of RTE challenges. The main task in these challenges has traditionally been to determine, given a text-hypothesis pair (T,H), whether T entails H. Discourse played no role in the first two RTE challenges as T's were constructed of short simplified texts. In RTE-3 (Giampiccolo et al., 2007), where some paragraph-long texts were included, inter-sentential relations became relevant for correct inference. Yet the texts in the task were manually modified to ensure they are self-contained. Consequently, little effort was invested by the challenges' participants to address discourse issues beyond the standard substitution of coreferring nominal phrases, using publicly available tools such as JavaRap (Qiu et al., 2004) or OpenNLP[1], e.g. (Bar-Haim et al., 2008).

A major step in the RTE challenges towards a more practical setting of text processing applications occurred with the introduction of the Search task in the Fifth RTE challenge (RTE-5). In this task entailing sentences are situated within documents and depend on other sentences for their correct interpretation. Thus, discourse becomes a substantial factor impacting inference. Surprisingly, discourse hardly received any treatment in this task beyond the standard use of coreference resolution (Castillo, 2009; Litkowski, 2009), and an attempt to address globally-known information by removing from H words that appear in document headlines (Clark and Harrison, 2009).

3 The RTE Search Task

The RTE-5 Search task was derived from the TAC Summarization task[2]. The dataset consists of several corpora, each comprised of news articles concerning a specific *topic*, such as the impact of global warming on the Arctic or the London terrorist attacks in 2005. *Hypotheses* were manually generated based on Summary Content Units (Nenkova et al., 2007), clause-long statements taken from manual summaries of the corpora. *Texts* are unmodified sentences in the articles. Given a topic and a hypothesis, entailment systems are required to identify all sentences in the topic's corpus that entail the hypothesis.

Each sentence-hypothesis pair in both the development and test sets was annotated, judging whether the sentence entails the hypothesis. Out of 20,104 annotations in the development set, only 810 were judged as positive. This small ratio (4%) of positive examples, in comparison to 50% in traditional RTE tasks, better corresponds to the natural distribution of entailing texts in a corpus, thus better simulates practical settings.

The task may seem as a variant of information retrieval (IR), as it requires finding specific texts in a corpus. Yet, it is fundamentally different from IR for two reasons. First, the target output is a set

[1] http://opennlp.sourceforge.net

[2] http://www.nist.gov/tac/2009/Summarization/

of sentences, each evaluated independently, rather than a set of documents. Second, the decision criterion is *entailment* rather than *relevance*.

Despite the above, apparently, IR techniques provided hard-to-beat baselines for the RTE Search task (MacKinlay and Baldwin, 2009), outperforming every other system that relied on inference without IR-based pre-filtering. At the current state of performance of entailment systems, it seems that lexical coverage largely overshadows any other approach in this task. Still, most (6 out of 8) participants in the challenge applied their entailment systems to the entire dataset without a prior retrieval of candidate sentences. F_1 scores for such systems vary between 10% and 33%, in comparison to over 40% of the IR-based methods.

4 The Baseline RTE System

In this work we used BIUTEE, Bar-Ilan University Textual Entailment Engine (Bar-Haim et al., 2008; Bar-Haim et al., 2009), a state of the art RTE system, as a baseline and as a basis for our discourse-based enhancements. This section describes this system's architecture; the methods by which it was augmented to address discourse are presented in Section 5.

To determine entailment, BIUTEE performs the following main steps:

Preprocessing First, all documents are parsed and processed with standard tools for named entity recognition (Finkel et al., 2005) and coreference resolution. For the latter purpose, we use OpenNLP and enable the substitution of coreferring terms. This is the only way by which BIUTEE addresses discourse, representing the state of the art in entailment systems.

Entailment-based transformations Given a T-H pair (both represented as dependency parse trees), the system applies a sequence of knowledge-based entailment transformations over T, generating a set of texts which are entailed by it. The goal is to obtain consequent texts which are more similar to H. Based on preliminary results on the development set, in our experiments (Section 6) we use WordNet (Fellbaum, 1998) as the system's only knowledge resource, using its synonymy, hyponymy and derivation relations.

Classification A supervised classifier, trained on the development set, is applied to determine entailment of each pair based on a set of syntactic and lexical syntactic features assessing the degree by which T and its consequents cover H.

5 Addressing Discourse

In the following subsections we describe the prominent discourse phenomena that affect inference, which we have identified in an analysis of the development set and addressed in our implementation. As mentioned, these phenomena are poorly addressed by available reference resolvers or fall completely out of their scope.

5.1 Augmented coreference set

A large number of coreference relations are comprised of terms which share lexical elements, (e.g. *"airliners's first flight"* and *"Airbus A380's first flight"*). Although common in coreference relations, standard resolvers miss many of these cases. For the purpose of identifying additional coreferring terms, we consider two noun phrases in the same document as coreferring if: (i) their heads are identical and (ii) no semantic incompatibility is found between their modifiers. The types of incompatibility we handle are: (a) mismatching numbers, (b) antonymy and (c) co-hyponymy (coordinate terms), as specified by WordNet. For example, two nodes of the noun *distance* would be considered incompatible if one is modified by *short* and the second by its antonym *long*. Similarly, two modifier co-hyponyms of *distance*, such as *walking* and *running* would also result such an incompatibility. Adding more incompatibility types (e.g. *first* vs. *second flight*) may further improve the precision of this method.

5.2 Global information

Key terms or prominent pieces of information that appear in the document, typically at the title or the first few sentences, are many times perceived as "globally" known throughout the document. For example, the geographic location of the document theme, mentioned at the beginning of the document, is assumed to be known from that point on, and will often not be mentioned explicitly in further sentences. This is a bridging phenomenon

that is typically not addressed by available discourse processing tools. To compensate for that, we identify key terms for each document based on *tf-idf* scores and consider them as global information for that document. For example, global terms for the topic discussing the ice melting in the Arctic, typically contain a location such as *Arctic* or *Antarctica* and terms referring to *ice*, like *permafrost* or *iceshelf*.

We use a variant of *tf-idf*, where term frequency is computed as follows: $tf(t_{i,j}) = n_{i,j} + \vec{\lambda}^{\top} \cdot \vec{f_{i,j}}$. Here, $n_{i,j}$ is the frequency of term i in document j ($t_{i,j}$), which is incremented by additional positive weights ($\vec{\lambda}$) for a set of features ($\vec{f_{i,j}}$) of the term. Based on our analysis, we defined the following features, which correlated mostly with global information: (i) does the term appear in the title? (ii) is it a proper name? (iii) is it a location? The weights for these features are set empirically.

The document's top-n global terms are added to each of its sentences. As a result, a global term that occurs in the hypothesis is matched in each sentence of the document, regardless of whether the term explicitly appears in the sentence.

Considering the previous sentence Another method for addressing missing coreference and bridging relations is based on the assumption that adjacent sentences often refer to the same entities and events. Thus, when extracting classification features for a given sentence, in addition to the features extracted from the parse tree of the sentence itself, we extract the same set of features from the current and previous sentences together. Recall the example presented in Section 1. T is annotated as entailing the hypothesis *"The AS-28 mini-submarine was trapped underwater"*, but the word *submarine*, e.g., appears only in its preceding sentence T'. Thus, considering both sentences together when classifying T increases its coverage of the hypothesis. Indeed, a bridging reference relates *on board* in T with *submarine* in T', justifying our assumption in this case.

5.3 Document-level classification

Beyond discourse references addressed above, further information concerning discourse and document structure is available in the Search setting and may contribute to entailment classification. We observed that entailing sentences tend to come in bulks. This reflects a common coherence aspect, where the discussion of a specific topic is typically continuous rather than scattered across the entire document. This *locality* phenomenon may be useful for entailment classification since knowing that a sentence entails the hypothesis increases the probability that adjacent sentences entail the hypothesis as well.

To capture this phenomenon, we use a two-phase meta-classification scheme, in which a *meta-classifier* utilizes entailment classifications of the first classification phase to extract *meta-features* and determine the final classification decision. This scheme also provides a convenient way to combine scores from multiple classifiers used in the first classification phase. We refer to these as *base-classifiers*. This scheme and the meta-features we used are detailed hereunder.

Let us write (s, h) for a sentence-hypothesis pair. We denote the set of pairs in the development (training) set as $\mathcal{D}$ and in the test set as $\mathcal{T}$. We split $\mathcal{D}$ into two halves, $\mathcal{D}_1$ and $\mathcal{D}_2$. We make use of n base-classifiers, $C_1, \ldots, C_n$, among which $C^{\star}$ is a designated classifier with additional roles in the process, as described below. Classifiers may differ, for example, in their classification algorithm. An additional meta-classifier is denoted C_M. The classification scheme is shown as Algorithm 1.

Algorithm 1 Meta-classification

Training

1: Extract features for every (s, h) in $\mathcal{D}$
2: Train $C_1, \ldots, C_n$ on $\mathcal{D}_1$
3: Classify $\mathcal{D}_2$, using $C_1, \ldots, C_n$
4: Extract meta-features for $\mathcal{D}_2$ using the classification of $C_1, \ldots, C_n$
5: Train C_M on $\mathcal{D}_2$

Classification

6: Extract features for every (s, h) in $\mathcal{T}$
7: Classify $\mathcal{T}$ using $C_1, \ldots, C_n$
8: Extract meta-features for $\mathcal{T}$
9: Classify $\mathcal{T}$ using C_M

At Step 1, features are extracted for every (s, h) pair in the training set, as in the baseline system.

In Steps 2 and 3 we split the training set into two halves (taking half of each topic), train n different classifiers on the first half and then classify the second half using each of the n classifiers. Given the classification scores of the n base-classifiers to the (s, h) pairs in the second half of the training set, $\mathcal{D}_2$, we add in Step 4 the meta-features described in Section 5.3.1.

After adding the meta-features, we train (Step 5) a meta-classifier on this new set of features. Test sentences then go through the same process: features are extracted for them and they are classified by the already trained n classifiers (Steps 6 and 7), meta-features are extracted in Step 8, and a final classification decision is made by the meta-classifier in Step 9.

A retrieval step may precede the actual entailment classification, allowing the processing of fewer and potentially "better" candidates.

5.3.1 Meta-features

The following features are extracted in our meta-classification scheme:

Classification scores The classification score of each of the n base-classifiers.

Title entailment In many texts, and in news articles in particular, the title and the first few sentences often represent the entire document's content. Thus, knowing whether these sentences entail the hypothesis may be an indicator to the general potential of the document to include entailing sentences. Two binary features are added according to the classification of $C^\star$ indicating whether the title entails the hypothesis and whether the first sentence entails it.

Second-closest entailment Considering the locality phenomenon described above, we add a feature assigning higher scores to sentences in the vicinity of an entailment environment. This feature is computed as the distance to the second-closest entailing sentence in the document (counting the sentence itself as well), according to the classification of $C^\star$. Formally, let i be the index of the current sentence and $\mathcal{J}$ be the set of indices of entailing sentences in the document according to $C^\star$. For each $j \in \mathcal{J}$ we compute $d_{i,j} = |i-j|$, and choose the second smallest $d_{i,j}$ as d_i. The idea is that if entailing sentences indeed always come in bulks, then $d_i = 1$ for all entailing sentences, but $d_i > 1$ for all non-entailing ones. Figure 1 illustrates such a case, comparing the *second-closest* distance with the distance to the *closest* entailing sentence. In the *closest* scheme we do not count the sentence as closest to itself since it would disregard the environment of the sentence altogether, eliminating the desired effect. We scale the distance and add the feature score: $-\log(d_i)$.

#	Ent?	Closest	d	2nd closest	d
1	NO	6	5	7	6
2	NO	6	4	7	5
3	NO	6	3	7	4
4	NO	6	2	7	3
5	NO	6	1	7	2
6	YES	7	1	7	1
7	YES	6 or 8	1	6 or 8	1
8	YES	7 or 9	1	7 or 9	1
9	YES	8	1	8	1
10	NO	8	1	8	2
11	NO	8	2	8	3

Figure 1: Comparison of the *closest* and *second-closest* schemes when applied to a bulk of entailing sentences (in white) situated within a non-entailing environment (in gray). Unlike the *closest* one, the *second-closest* scheme assigns larger distance values to non-entailing sentences located on the 'edge' of the bulk (5 and 10) than to entailing ones.

Smoothed entailment This feature addressed the locality phenomenon by smoothing the classification score of sentence i with the scores of adjacent sentences, weighted by their distance from the current sentence i. Let $s(i)$ be the score assigned by $C^\star$ to sentence i. We add the Smoothed Entailment feature score:

$$\text{SE}(i) = \frac{\sum_w (b^{|w|} \cdot s(i+w))}{\sum_w (b^{|w|})}$$

where $0 < b < 1$ is the decay parameter and w is an integer bounded between $-N$ and N, denoting the distance from sentence *i*.

1st sentence entailing title Bensley and Hickl (2008) showed that the first sentence in a news article typically entails the article's title. We therefore assume that in each document, $s_1 \Rightarrow s_0$, where s_1 and s_0 are the document's first sentence and title respectively. Hence, under entailment transitivity, if $s_0 \Rightarrow h$ then $s_1 \Rightarrow h$. The corresponding binary feature states whether the sentence being classified is the document's first sentence *and* the title entails h according to $C^\star$.

	P (%)	R (%)	F_1 (%)
BIU-BL	14.53	55.25	23.00
BIU-DISC	20.82	57.25	30.53
BIU-BL^3	14.86	59.00	23.74
BIU-DISC$_{no-loc}$	22.35	57.12	32.13
All-yes baseline	4.6	100.0	8.9

Table 1: Micro-average results.

	P (%)	R (%)	F_1 (%)
By Topic			
BIU-BL	16.54	55.62	25.50
BIU-DISC	22.69	57.96	32.62
All-yes baseline	4.85	100.00	9.25
By Hypothesis			
BIU-BL	22.87	59.62	33.06
BIU-DISC	27.81	61.97	38.39
All-yes baseline	4.96	100.00	9.46

Table 2: Macro-average results.

Note that the above locality-based features rely on high accuracy of the base classifier $C^\star$. Otherwise, it will provide misleading information to the features computation. We analyze the effect of this accuracy in Section 6.

6 Results and Analysis

Using the RTE-5 Search data, we compare BIUTEE in its baseline configuration (cf. Section 4), denoted *BIU-BL*, with its discourse-aware enhancement (*BIU-DISC*) which uses all the components described in Section 5. To alleviate the strong IR effect described in Section 3, both systems are applied to the complete datasets (both training and test), without candidates pre-filtering.

BIU-DISC uses three base-classifiers ($n = 3$): *SVM*^{perf} (Joachims, 2006), and Naïve Bayes and Logistic Regression from the WEKA package (Witten and Frank, 2005). The first among these is set as our designated classifier $C^\star$, which is used for the computation of the document-level features. *SVM*^{perf} is also used for the meta-classifier. For the smoothed entailment score (cf. Section 5.3), we used $b = 0.9$ and $N = 3$. Global information is added by enriching each sentence with the highest-ranking term in the document, according to *tf-idf* scores (cf. Section 5.2), where document frequencies were computed based on about half a million documents from the TIPSTER corpus (Harman, 1992). The set of weights $\vec{\lambda}$ equals $\{2, 1, 4\}$ for title terms, proper names and locations, respectively. All parameters were tuned based on a 10-fold cross-validation on the development set, optimizing the micro-averaged F_1.

The results are presented in Table 1. As can be seen in the table, *BIU-DISC* outperforms *BIU-BL* in every measure, showing the impact of addressing discourse in this setting. To rule out the option that the improvement is simply due to the fact that we use three classifiers for *BIU-DISC* and a single one for *BIU-BL*, we show (*BIU-BL*^{3}) the results when the baseline system is applied in the same meta-classification configuration as *BIU-DISC*, with the same three classifiers. Apparently, without the discourse information this configuration's contribution is limited.

As mentioned in Section 5.3, the benefit from the locality features rely directly on the performance of the base classifiers. Hence, considering the low precision scores obtained here, we applied *BIU-DISC* to the data in the meta-classification scheme, but with locality features removed. The results, shown as *BIU-DISC*$_{no-loc}$ in the Table, indicate that indeed performance increases without these features. The last line of the table shows the results obtained by a naïve baseline where all test-set pairs are considered entailing.

For completeness, Table 2 shows the macro-averaged results, when averaged over the topics or over the hypotheses. Although we tuned our system to maximize micro-averaged F_1, these figures comply with the ones shown in Table 1.

Analysis of locality As discussed in Section 5, determining whether a sentence entails a hypothesis should take into account whether adjacent sentences also entail the hypothesis. In the above experiment we were unable to show the contribution of our system's component that attempts to capture this information; on the contrary, the results show it had a negative impact on performance.

Still, we claim that this information can be useful when used within a more accurate system. We try to validate this conjecture by understanding how performance of the locality features varies as the systems becomes more accurate. We do so via the following simulation.

When classifying a certain sentence, the classi-

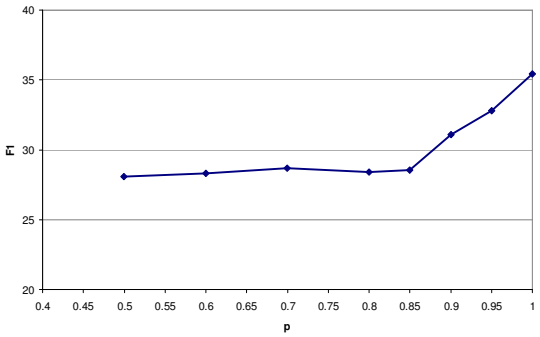


Figure 2: F_1 performance of *BIU-DISC* as a function of the accuracy in classifying adjacent sentences.

fications of its adjacent sentences are given by an *oracle classifier* that provides the correct answer with probability p. The system is applied using two locality features: the *1st sentence entailing title* feature and a close variant of the *smoothed entailment* feature, which calculates the weighted average of adjacent sentences, but disregards the score of the currently evaluated sentence.[3] Thus we supply information about adjacent sentences and test whether overall performance increases with the accuracy of this information.

We performed this experiment for p in a range of [0.5-1.0]. Figure 2 shows the results of this simulation, based on the average F_1 of five runs for each p. Since performance, from a certain point, increases with the accuracy of the oracle classifier, we can conclude that indeed precise information about adjacent sentences improves performance on the current sentence, and that locality is a true phenomenon in the data. We note, however, that performance improves only when accuracy is very high, suggesting the currently limited practical potential of this information, at least in the way locality was represented in this work.

Ablation tests Table 3 presents the results of the ablation tests performed to evaluate the contribution of each component. Based on the result reported in Table 1 and the above discussion, the tests were performed relative to *BIU-DISC*$_{no-loc}$, the optimal configuration. As seen in the table, the removal of each component causes a drop in results. For global information we see a minor difference, which is not surprising considering the conservative approach we took, using a single global term for each sentence. Possibly, this information is also included in the other components, thus proving no marginal contribution relative to them. Under the conditions of an overwhelming majority of negative examples, this is a risky method to use, and should be considered when the ratio of positive examples is higher. For future work, we intend to use this information via classification features (e.g. the coverage obtained with and without global information), rather than the crude addition of the term to the sentence.

[3]The *second-closest entailment* feature was not used as it considers the oracle's decision for the current sentence, while we wish to use only information about adjacent sentences.

Component removed	F_1 (%)	ΔF_1 (%)
Previous sent. features	28.55	3.58
Augmented coref.	26.73	5.40
Global information	31.76	0.37

Table 3: Results of ablation tests relative to *BIU-DISC*$_{no-loc}$. The columns specify the component removed, the micro-averaged F_1 score achieved without it, and the marginal contribution of the component.

Analysis of augmented coreferences We analyzed the performance of the component for augmenting coreference relations relative to the OpenNLP resolver. Recall that our component works on top of the resolver's output and can add or remove coreference relations. As a complete annotation of coreference chains in the dataset is unavailable, we performed the following evaluation. Recall is computed based on the number of identified pairs from a sample of 100 intra-document coreference and bridging relations from the annotated dataset described in (Mirkin et al., 2010). Precision is computed based on 50 pairs sampled from the output of each method, equally distributed over topics. The results, shown in Table 4, indicate the much higher recall obtained by our component at some cost in precision. Although rather simple, the ablation test of this component shows its usefulness. Still, both methods achieve low absolute recall, suggesting the need for more robust tools for this task.

	P (%)	R (%)	F_1 (%)
OpenNLP	74	16	26.3
Augmented coref.	60	28	38.2

Table 4: Performance of coreference methods.

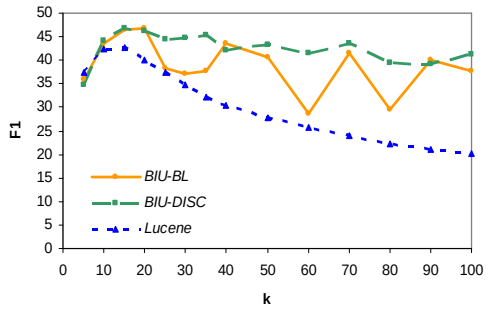


Figure 3: F_1 performance as a function of the number of retrieved candidates.

Candidate retrieval setting As mentioned in Section 3, best performance of RTE systems in the task was obtained when applying a first step of IR-based candidate filtering. We therefore compare the performance of *BIU-DISC* with that of *BIU-BL* under this setting as well.[4] For candidate retrieval we used Lucene, a state of the art search engine[5], in a range of top-k retrieved candidates. The results are shown in Figure 3. For reference, the figure also shows the performance along this range of Lucene as-is, when no further inference is applied to the retrieved candidates.

While *BIU-DISC* does not outperform *BIU-BL* at every point, the area under the curve is clearly larger for *BIU-DISC*. The figure also indicates that *BIU-DISC* is far more robust, maintaining a stable F_1 and enabling a stable tradeoff between recall and precision along the whole range (recall ranges between 42% and 55% for $k \in [15 - 100]$, with corresponding precision range of 51% to 33%).

Finally, Table 5 shows the results of the best systems as determined in our first experiment. We performed a single experiment to compare *BIU-DISC*$_{no-loc}$ and *BIU-BL*^{3} under a candidate retrieval setting, using $k = 20$, where both systems highly perform. We compare these results to the highest score obtained by Lucene, as well as to the two best submissions to the RTE-5 Search task[6]. *BIU-DISC*$_{no-loc}$ outperforms all other methods and its result is significantly better than *BIU-BL*^{3} with $p < 0.01$ according to McNemar's test.

[4]This time, for global information, the document's three highest ranking terms were added to each sentence.

[5]http://lucene.apache.org

[6]The best one is an earlier version of this work (Mirkin et al., 2009); the second is MacKinlay and Baldwin's (2009).

	P (%)	R (%)	F_1 (%)
BIU-DISC$_{no-loc}$	50.77	45.12	47.78
BIU-BL^3	51.68	40.38	45.33
Lucene, top-15	35.93	52.50	42.66
RTE-5 best	40.98	51.38	45.59
RTE-5 second-best	42.94	38.00	40.32

Table 5: Performance of best configurations.

7 Conclusions

While it is generally assumed that discourse interacts with semantic entailment inference, the concrete impacts of discourse on such inference have been hardly explored. This paper presented a first empirical investigation of discourse processing aspects related to entailment. We argue that available discourse processing tools should be substantially improved towards this end, both in terms of the phenomena they address today, namely nominal coreference, and with respect to the covering of additional phenomena, such as bridging anaphora. Our experiments show that even rather simple methods for addressing discourse can have a substantial positive impact on the performance of entailment inference. Concerning the locality phenomenon stemming from discourse coherence, we learned that it does carry potentially useful information, which might become beneficial in the future when better-performing entailment systems become available. Until then, integrating this information with entailment confidence may be useful. Overall, we suggest that entailment systems should extensively incorporate discourse information, while developing sound algorithms for addressing various discourse phenomena, including the ones described in this paper.

Acknowledgements

The authors are thankful to Asher Stern and Ilya Kogan for their help in implementing and evaluating the augmented coreference component, and to Roy Bar-Haim for useful advice concerning this paper. This work was partially supported by the Israel Science Foundation grant 1112/08 and the PASCAL-2 Network of Excellence of the European Community FP7-ICT-2007-1-216886. Jonathan Berant is grateful to the Azrieli Foundation for the award of an Azrieli Fellowship.

References

Bar-Haim, Roy, Jonathan Berant, Ido Dagan, Iddo Greental, Shachar Mirkin, Eyal Shnarch, and Idan Szpektor. 2008. Efficient semantic deduction and approximate matching over compact parse forests. In *Proc. of Text Analysis Conference (TAC)*.

Bar-Haim, Roy, Jonathan Berant, and Ido Dagan. 2009. A compact forest for scalable inference over entailment and paraphrase rules. In *Proc. of EMNLP*.

Bensley, Jeremy and Andrew Hickl. 2008. Unsupervised resource creation for textual inference applications. In *Proc. of LREC*.

Bentivogli, Luisa, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, Medea Lo Leggio, and Bernardo Magnini. 2009a. Considering discourse references in textual entailment annotation. In *Proc. of the 5th International Conference on Generative Approaches to the Lexicon (GL2009)*.

Bentivogli, Luisa, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. 2009b. The fifth PASCAL recognizing textual entailment challenge. In *Proc. of TAC*.

Castillo, Julio J. 2009. Sagan in TAC2009: Using support vector machines in recognizing textual entailment and TE search pilot task. In *Proc. of TAC*.

Clark, Peter and Phil Harrison. 2009. An inference-based approach to recognizing entailment. In *Proc. of TAC*.

Clark, Herbert H. 1975. Bridging. In Schank, R. C. and B. L. Nash-Webber, editors, *Theoretical issues in natural language processing*, pages 169–174. Association of Computing Machinery.

Dagan, Ido, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges*, volume 3944 of *Lecture Notes in Computer Science*, pages 177–190. Springer.

Dagan, Ido, Bill Dolan, Bernardo Magnini, and Dan Roth. 2009. Recognizing textual entailment: Rational, evaluation and approaches. *Natural Language Engineering*, pages 15(4):1–17.

Dali, Lorand, Delia Rusu, Blaz Fortuna, Dunja Mladenic, and Marko Grobelnik. 2009. Question answering based on semantic graphs. In *Proc. of the Workshop on Semantic Search (SemSearch 2009)*.

Fellbaum, Christiane, editor. 1998. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press.

Finkel, Jenny Rose, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proc. of ACL*.

Giampiccolo, Danilo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proc. of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.

Harabagiu, Sanda and Andrew Hickl. 2006. Methods for using textual entailment in open-domain question answering. In *Proc. of ACL*.

Harman, Donna. 1992. The DARPA TIPSTER project. *SIGIR Forum*, 26(2):26–28.

Joachims, Thorsten. 2006. Training linear SVMs in linear time. In *Proc. of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*.

Li, Fangtao, Yang Tang, Minlie Huang, and Xiaoyan Zhu. 2009. Answering opinion questions with random walks on graphs. In *Proc. of ACL-IJCNLP*.

Litkowski, Ken. 2009. Overlap analysis in textual entailment recognition. In *Proc. of TAC*.

MacKinlay, Andrew and Timothy Baldwin. 2009. A baseline approach to the RTE5 search pilot. In *Proc. of TAC*.

Mirkin, Shachar, Roy Bar-Haim, Jonathan Berant, Ido Dagan Eyal Shnarch, Asher Stern, and Idan Szpektor. 2009. Addressing discourse and document structure in the RTE search task. In *Proc. of TAC*.

Mirkin, Shachar, Ido Dagan, and Sebastian Padó. 2010. Assessing the role of discourse references in entailment inference. In *Proc. of ACL*.

Nenkova, Ani, Rebecca Passonneau, and Kathleen Mckeown. 2007. The pyramid method: incorporating human content selection variation in summarization evaluation. In *ACM Transactions on Speech and Language Processing*.

Qiu, Long, Min-Yen Kan, and Tat-Seng Chua. 2004. A public reference implementation of the RAP anaphora resolution algorithm. In *Proc. of LREC*.

Romano, Lorenza, Milen Kouylekov, Idan Szpektor, and Ido Dagan. 2006. Investigating a generic paraphrase-based approach for relation extraction. In *Proc. of EACL*.

Witten, Ian H. and Eibe Frank. 2005. *Data Mining: Practical machine learning tools and techniques, 2nd Edition*. Morgan Kaufmann, San Francisco.

Evaluating Dependency Representation for Event Extraction

Makoto Miwa[1] **Sampo Pyysalo**[1] **Tadayoshi Hara**[1] **Jun'ichi Tsujii**[1,2,3]
[1]Department of Computer Science, the University of Tokyo
[2]School of Computer Science, University of Manchester
[3]National Center for Text Mining
{mmiwa,smp,harasan,tsujii}@is.s.u-tokyo.ac.jp

Abstract

The detailed analyses of sentence structure provided by parsers have been applied to address several information extraction tasks. In a recent bio-molecular event extraction task, state-of-the-art performance was achieved by systems building specifically on dependency representations of parser output. While intrinsic evaluations have shown significant advances in both general and domain-specific parsing, the question of how these translate into practical advantage is seldom considered. In this paper, we analyze how event extraction performance is affected by parser and dependency representation, further considering the relation between intrinsic evaluation and performance at the extraction task. We find that good intrinsic evaluation results do not always imply good extraction performance, and that the types and structures of different dependency representations have specific advantages and disadvantages for the event extraction task.

1 Introduction

Advanced syntactic parsing methods have been shown to effective for many information extraction tasks. The BioNLP 2009 Shared Task, a recent bio-molecular event extraction task, is one such task: analysis showed that the application of a parser correlated with high rank in the task (Kim et al., 2009). The automatic extraction of biomolecular events from text is important for a number of advanced domain applications such as pathway construction, and event extraction thus a key task in Biomedical Natural Language Processing (BioNLP).

Methods building feature representations and extraction rules around dependency representations of sentence syntax have been successfully applied to a number of tasks in BioNLP. Several parsers and representations have been applied in high-performing methods both in domain studies in general and in the BioNLP'09 shared task in particular, but no direct comparison of parsers or representations has been performed. Likewise, a number of evaluation of parser outputs against gold standard corpora have been performed in the domain, but the broader implications of the results of such intrinsic evaluations are rarely considered. The BioNLP'09 shared task involved documents contained also in the GENIA treebank (Tateisi et al., 2005), creating an opportunity for direct study of intrinsic and task-oriented evaluation results. As the treebank can be converted into various dependency formats using existing format conversion methods, evaluation can further be extended to cover the effects of different representations.

In this this paper, we consider three types of dependency representation and six parsers, evaluating their performance from two different aspects: dependency-based intrinsic evaluation, and effectiveness for bio-molecular event extraction with a state-of-the-art event extraction system. Comparison of intrinsic and task-oriented evaluation re-

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 779–787,
Beijing, August 2010

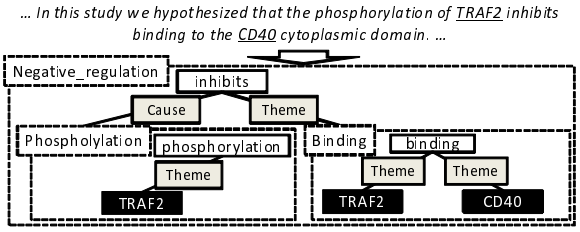


Figure 1: Event Extraction.

sults shows that performance against gold standard annotations is not always correlated with event extraction performance. We further find that the dependency types and overall structures employed by the different dependency representations have specific advantages and disadvantages for the event extraction task.

2 Bio-molecular Event Extraction

In this study, we adopt the event extraction task defined in the BioNLP 2009 Shared Task (Kim et al., 2009) as a model information extraction task. Figure 1 shows an example illustrating the task of event extraction from a sentence. The shared task provided common and consistent task definitions, data sets for training and evaluation, and evaluation criteria. The shared task defined five simple events (Gene expression, Transcription, Protein catabolism, Phosphorylation, and Localization) that take one core argument, a multi-participant binding event (Binding), and three regulation events (Regulation, Positive regulation, and Negative regulation) used to capture both biological regulation and general causation. The participants of simple and Binding events were specified to be of the general Protein type, while regulation-type events could also take other events as arguments, creating complex event structures.

We consider two subtasks, Task 1 and Task 2, out of the three defined in the shared task. Task 1 focuses on core event extraction, and Task 2 involves augmenting extracted events with secondary arguments (Kim et al., 2009). Events are represented with a textual trigger, type, and arguments, where the trigger is a span of text that states the event in text. In Task 1 the event arguments that need to be extracted are restricted to the core Theme and Cause roles, with secondary arguments corresponding to locations and sites considered in Task 2.

2.1 Event Extraction System

For evaluation, we apply the system of Miwa et al. (2010b). The system was originally developed for finding core events (Task 1) using the native output of the Enju and GDep parsers. The system consists of three supervised classification-based modules: a trigger detector, an event edge detector, and a complex event detector. The trigger detector classifies each word into the appropriate event types, the event edge detector classifies each edge between an event and a candidate participant into an argument type, and the complex event detector classifies event candidates constructed by all edge combinations, deciding between event and non-event. The system uses one-vs-all support vector machines (SVMs) for classification.

The system operates on one sentence at a time, building features for classification based on the syntactic analyses for the sentence provided by the two parsers as well as the sequence of the words in the sentence, including the target candidate. The features include the constituents/words around entities (triggers and proteins), the dependencies, and the shortest paths among the entities. The feature generation is format-independent regarding the shared properties of different formats, but makes use also of format-specific information when available for extracting features, including the dependency tags, word-related information (e.g. a lexical entry in Enju format), and the constituents and their head information.

We apply here a variant of the base system incorporating a number of modifications. The applied system performs feature selection removing two classes of features that were found not to be beneficial for extraction performance, and applies a refinement of the trigger expressions of events. The system is further extended to find also secondary arguments (Task 2). For a detailed description of these improvements, we refer to Miwa et al. (2010a).

3 Parsers and Representations

Six publicly available parsers and three dependency formats are considered in this paper. The

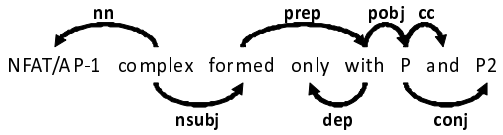


Figure 2: Stanford basic dependency tree

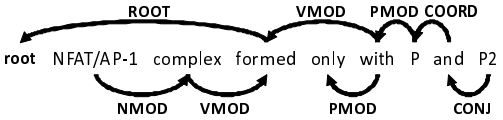


Figure 3: CoNLL-X dependency tree

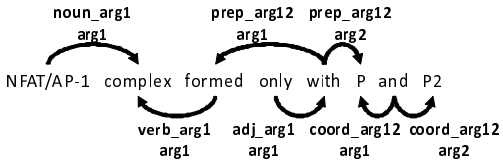


Figure 4: Predicate Argument Structure

parsers are GDep (Sagae and Tsujii, 2007), the Bikel parser (Bikel) (Bikel, 2004), the Stanford parser with two probabilistic context-free grammar (PCFG) models[1] (Wall Street Journal (WSJ) model (Stanford WSJ) and "augmented English" model (Stanford eng)) (Klein and Manning, 2003), the Charniak-Johnson reranking parser, using David McClosky's self-trained biomedical parsing model (MC) (McClosky, 2009), the C&C CCG parser, adapted to biomedical text (C&C) (Rimell and Clark, 2009), and the Enju parser with the GENIA model (Miyao et al., 2009). The formats are Stanford Dependencies (SD) (Figure 2), the CoNLL-X dependency format (CoNLL) (Figure 3) and the predicate-argument structure (PAS) format used by Enju (Figure 4). With the exception of Stanford and Enju, the analyses of these parsers were provided by the BioNLP 2009 Shared Task organizers.

The six parsers operate in a number of different frameworks, reflected in their analyses. GDep is a native dependency parser that produces CoNLL dependency trees, with dependency types similar to those of CoNLL 2007. Bikel, Stanford, and MC

[1]Experiments showed no benefit from using the lexicalized models with the Stanford parser.

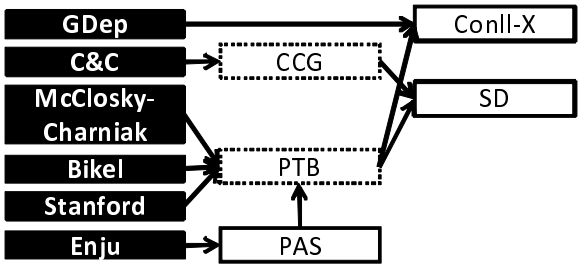


Figure 5: Format conversion dependencies in six parsers. Formats adopted for the evaluation are shown in solid boxes. SD: Stanford Dependency format, CCG: Combinatory Categorial Grammar output format, PTB: Penn Treebank format, and PAS: Predicate Argument Structure in Enju format.

are phrase-structure parsers trained on Penn Treebank format (PTB) style treebanks, and they produce PTB trees. C&C is a deep parser based on Combinatory Categorial Grammar (CCG), and its native output is in a CCG-specific format. The output of C&C can be converted into SD by a rule-based conversion script (Rimell and Clark, 2009). Enju is deep parser based on Head-driven Phrase Structure Grammar (HPSG) and produces a format containing predicate argument structures along with a phrase structure tree in Enju format, which can be converted into PTB format (Miyao et al., 2009).

For direct comparison and for the study of contribution of the formats in which the six parsers output their analyses to task performance, we apply a number of conversions between the outputs, shown in Figure 5. The Enju PAS output is converted into PTB using the method introduced by (Miyao et al., 2009). SD is generated from PTB by the Stanford tools (de Marneffe et al., 2006), and CoNLL generated from PTB by using Treebank Converter (Johansson and Nugues, 2007). With the exception of GDep, all CoNLL outputs are generated by the conversion and thus share dependency types. We note that all of these conversions can introduce some errors in the conversion process.

4 Evaluation Setting

4.1 Event Extraction Evaluation

Event extraction performance is evaluated using the evaluation script provided by the BioNLP'09 shared task organizers for the development data set, and the online evaluation system of the task for the test data set[2] . Results are reported under the official evaluation criterion of the task, i.e. the "Approximate Span Matching/Approximate Recursive Matching" criterion.

The event extraction system described in Section 2.1 is used with the default settings given in (Miwa et al., 2010b). The C-values of SVMs are set to 1.0, but the positive and negative examples are balanced by placing more weight on the positive examples. The examples predicted with confidence greater than 0.5, as well as the examples with the most confident labels, are extracted. Task 1 and Task 2 are solved at once for the evaluation.

Some of the parse results do not include word base forms or part-of-speech (POS) tags, which are required by the event extraction system. To apply these parsers, the GENIA Tagger (Tsuruoka et al., 2005) output is adopted to add this information to the results.

4.2 Dependency Representation Evaluation

The parsers are evaluated with precision, recall, and F-score for each dependency type. We note that the parsers may produce more fine-grained word segmentations than that of the gold standard: for example, two words "p70(S6)-kinase activation" in the gold standard tree (Figure 6 (a)) is segmented into five words by Enju (Figure 6 (b)). In the evaluation the word segmentations in the gold tree are used, and dependency transfer and word-based normalization are performed to match parser outputs to these. Dependencies related to the segmentations are transferred to the enclosing word as follows. If one word is segmented into several segments by a parser, all the dependencies between the segments are removed (Figure 6 (c)) and the dependency between another word and the segments is converted into the dependency between the two words (Figure 6 (d)).

The parser outputs in SD and CoNLL can be assumed to be trees, so each node in the tree have only one parent node. However, in the converted tree nodes can have more than one parent. We cannot simply apply accuracy, or (un)labeled attachment score[3]. Word-based normalization is performed to avoid negative impact by the word segmentations by parsers. When (a) and (d) in Figure 6 are compared, the counts of correct relations will be 1.0 (0.5 for upper NMOD and 0.5 for lower NMOD in Figure 6 (d)) for the parser (precision), and the counts of correct relations will be 1.0 (for NMOD in Figure 6 (a)) for the gold (recall). This F-score is a good approximation of accuracy.

4.3 GENIA treebank processing

For comparison and evaluation, the texts in the GENIA treebank (Tateisi et al., 2005) are converted to the various formats as follows. To create PAS, the treebank is converted with Enju, and for trees that fail conversion, parse results are used instead. The GENIA treebank is also converted into PTB[4], and then converted into SD and CoNLL as described in Section 3. While based on manually annotated gold data, the converted treebanks are not always correct due to conversion errors.

5 Evaluation

This section presents evaluation results. Intrinsic evaluation is first performed in Section 5.1. Section 5.2 considers the effect of different SD variants. Section 5.3 presents the results of experiments with different parsers. Section 5.4 shows the performance of different parsers. Finally, the performance of the event extraction system is discussed in context of other proposed methods for the task in Section 5.5.

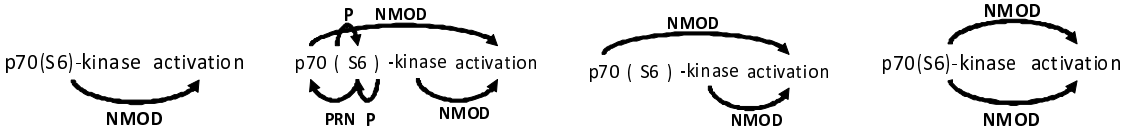
5.1 Intrinsic Evaluation

We initially briefly consider the results of an intrinsic evaluation comparing parser outputs to reference data automatically derived from the gold standard treebank. Table 1 shows results for the parsers whose outputs could be converted into the

[2] http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/SharedTask/

[3] http://nextens.uvt.nl/~conll/

[4] http://categorizer.tmit.bme.hu/~illes/genia_ptb/



(a) Gold Word Segmentations

(b) Parser Word Segmentations

(c) Inner Dependency Removal

(d) Dependency Transfer

Figure 6: Example of Word Segmentations of the words by gold and Enju and Dependency Transfer.

	Typed						Untyped					
	SD			CoNLL			SD			CoNLL		
	P	R	F	P	R	F	P	R	F	P	R	F
Bikel	70.31	70.37	70.34	77.81	77.56	77.69	80.54	80.60	80.57	82.43	82.18	82.31
SP WSJ	74.11	73.94	74.03	81.41	81.47	81.44	81.36	81.16	81.26	84.05	84.05	84.05
SP eng	79.08	78.89	78.98	84.92	84.82	84.87	84.16	83.96	84.06	86.54	86.47	86.51
C&C	80.31	78.04	79.16		-		84.91	82.28	83.57		-	
MC	79.56	79.63	79.60	88.13	87.87	88.00	87.43	87.50	87.47	89.81	89.42	89.62
Enju	85.59	85.62	85.60	88.59	89.51	89.05	88.28	88.30	88.29	90.24	90.77	90.50

Table 1: Comparison of precision, recall, and F-score results with five parsers (two models for Stanford) in two different formats on the development data set (SP abbreviates for Stanford Parser). Results shown separately for evaluation including dependency types and one eliminating them. Parser/model combinations above the line do not use in-domain data, others do.

SD and CoNLL dependency representations using the Stanford tools and Treebank Converter, respectively. For Stanford, both the Penn Treebank WSJ section and "augmented English" (eng) models were tested; the latter includes biomedical domain data. The Enju results for PAS are 91.48 with types and 93.39 without in F-score. GDep not shown as its output is not compatible with that of Treebank Converter.

Despite numerical differences, the two representations and two criteria (typed/untyped) all produce largely the same ranking of the parsers.[5] The evaluations also largely agree on the magnitude of the reduction in error afforded through the use of in-domain training data for the Stanford parser, with all estimates falling in the 15-19% range. Similarly, all show substantial differences between the parsers, indicating e.g. that the error rate of Enju is 50% or less of that of Bikel.

These results serve as a reference point for extrinsic evaluation results. However, it should be noted that as the parsers make use of annotated domain training data to different extents, this evaluation does not provide a sound basis for direct comparison of the parsers themselves.

[5]One larger divergence is between typed and untyped SD results for MC. Analysis suggest one cause is frequent errors in tagging hyphenated noun-modifiers such as *NF-kappaB* as adjectives.

	BD	CD	CDP	CTD
Task 1	**55.60**	54.35	54.59	54.42
Task 2	**53.94**	52.65	52.88	52.76

Table 2: Comparison of the F-score results with different SD variants on the development data set with the MC parser. The best score in each task is shown in bold.

5.2 Stanford Dependency Setting

SD have four different variants: basic dependencies (BD), collapsed dependencies (CD), collapsed dependencies with propagation of conjunct dependencies (CDP), and collapsed tree dependencies (CTD) (de Marneffe and Manning, 2008). Except for BD, these variants do not necessarily connect all the words in the sentence, and CD and CDP do not necessarily form a tree structure. Table 2 shows the comparison results with the MC parser. Dependencies are generalized by removing expressions after "_" of the dependencies (e.g.

"_with" in prep_with) for better performance. We find that basic dependencies give the best performance to event extraction, with little difference between the other variants. This result is surprising, as variants other than basic have features such as the resolution of conjunctions that are specifically designed for practical applications. However, basic dependencies were found to consistently provide best performance also for the other parsers[6]. Thus, in the following evaluation, the basic dependencies are adopted for all SD results.

5.3 Parser Comparison on Event Extraction

Results with different parsers and different formats on the development data set are summarized in Table 3. Baseline results are produced by removing dependency information from the parse results. The baseline results differ between the representations as the word base forms and POS tags produced by the GENIA tagger for use with SD and CoNLL are different from PAS, and because head word information in the Enju format is used. The evaluation finds best results for both tasks with Enju, using its native output format. However, as discussed in Section 2.1, the treatment of PAS and the other two formats are slightly different, this result does not necessarily indicate that PAS is the best alternative for event extraction.

The Bikel and Stanford WSJ parsers, lacking models adapted to the biomedical domain, performs mostly worse than the other parsers. The other parsers, even though trained on the treebank, do not provide performance as high as that for using the GENIA treebank, but, with the exception of Stanford eng with CoNLL, results with the parsers are only slightly worse than results with the treebank. The results with the data derived from the GENIA treebank can be considered as upper bounds for the parsers and formats at the task, although conversion errors are expected to lower these bounds to some extent. The results suggest that there is relative little remaining benefit to be gained from improving parser performance.

[6]Collapsed tree dependencies are not evaluated on the C&C parser since the conversion is not provided.

5.4 Effects of Dependency Representation

Intrinsic evaluation results (Section 5.1) cannot be used directly for comparing the parsers, since some of the parsers contain models trained on the GENIA treebank. To investigate the effects of the evaluation results to the event extraction, we performed event extraction with eliminating the dependency types. Table 4 summarizes the results with the dependency structures (without the dependency types) on the development data set. Interestingly, we find the performance increases in Bikel and Stanford by eliminating the dependency types. This implies that the inaccurate dependency types shown in Table 1 confused the event extraction system. SD and PAS drops more than CoNLL, and Enju with CoNLL structures perform best in total when the dependency types are removed. This result shows that the formats have their own strengths in finding events, and CoNLL structure with SD or PAS types can be a good representation for the event extraction.

By comparing Table 3, Table 1, and Table 4, we found that the better dependency performance does not always produce better event extraction performance especially when the difference of the dependency performance is small. MC and Enju results show that performance in dependency is important for event extraction. SD can be better than CoNLL for the event extraction (shown with the gold treebank data in Table 3), but the types and relations of CoNLL were well predicted, and MC and Enju performed better for CoNLL than for SD in total.

5.5 Performance of Event Extraction System

Several systems are compared by the extraction performance on the shared task test data in Table 5. GDep and Enju with PAS are used for the evaluation, which is the same evaluation setting with the original system by Miwa et al. (2010b). The performance of the best systems in the original shared task is shown for reference ((Björne et al., 2009) in Task 1 and (Riedel et al., 2009) in Task 2). The event extraction system performs significantly better than the best systems in the shared task, further outperforming the original system. This shows that the comparison of the parsers is performed with a state-of-the-art sys-

	Task 1			Task 2		
	SD	CoNLL	PAS	SD	CoNLL	PAS
Baseline	51.05	-	50.42	49.17	-	48.88
Bikel	53.29	53.22	-	51.40	51.27	-
Stanford WSJ	53.51	54.38	-	52.02	52.04	-
Stanford eng	55.02	53.66	-	53.41	52.74	-
GDep	-	55.70	-	-	54.37	-
MC	55.60	56.01	-	53.94	54.51	-
C&C	56.09	-	-	54.27	-	-
Enju	55.48	55.74	**56.57**	54.06	54.37	**55.31**
GENIA	56.34	56.09	57.94	55.04	54.57	56.40

Table 3: Comparison of F-score results with six parsers in three different formats on the development data set. Results without dependency information are shown as baselines. The results with the GENIA treebank (converted into PTB and PAS) are shown for comparison. The best score in each task is shown in bold, and the best score in each task and format is underlined.

	Task 1			Task 2		
	SD	CoNLL	PAS	SD	CoNLL	PAS
Bikel	53.41 (+0.12)	53.92 (+0.70)	-	51.59 (+0.19)	52.21 (+0.94)	-
Stanford WSJ	53.03 (-0.48)	54.52 (+0.14)	-	51.43 (-0.59)	52.60 (-0.14)	-
Stanford eng	54.48 (-0.54)	54.02 (+0.36)	-	52.88 (-0.53)	52.28 (+0.24)	-
GDep	-	54.97 (-0.73)	-	-	53.71 (-0.66)	-
MC	54.22 (-1.38)	55.24 (-0.77)	-	52.73 (-1.21)	53.42 (-1.09)	-
C&C	54.64 (-1.45)	-	-	52.98 (-1.29)	-	-
Enju	53.74 (-1.74)	**55.66** (-0.08)	55.23 (-1.34)	52.29 (-1.77)	**53.97** (-0.40)	53.69 (-1.62)
GENIA	55.79 (-0.55)	55.64 (-0.45)	56.42 (-1.52)	54.17 (-0.87)	53.83 (-0.74)	55.34 (-1.06)

Table 4: Comparison of F-score results with six parsers in three different dependency structures (without the dependency types) on the development data set. The changes from Table 3 are shown.

	Simple	Binding	Regulation	All
		Task 1		
Ours	**66.84** / **78.22** / **72.08**	48.70 / 52.65 / 50.60	**38.48** / **55.06** / **45.30**	**50.13** / **64.16** / **56.28**
Miwa	65.31 / 76.44 / 70.44	**52.16** / **53.08** / **52.62**	35.93 / 46.66 / 40.60	48.62 / 58.96 / 53.29
Björne	64.21 / 77.45 / 70.21	40.06 / 49.82 / 44.41	35.63 / 45.87 / 40.11	46.73 / 58.48 / 51.95
Riedel	N/A	23.05 / 48.19 / 31.19	26.32 / 41.81 / 32.30	36.90 / 55.59 / 44.35
Baseline	62.94 / 68.38 / 65.55	48.41 / 34.50 / 40.29	29.40 / 40.00 / 33.89	43.93 / 50.11 / 46.82
		Task 2		
Ours	**65.43** / **75.56** / **70.13**	**46.42** / **50.31** / **48.29**	**38.18** / **54.45** / **44.89**	**49.20** / **62.57** / **55.09**
Riedel	N/A	22.35 / 46.99 / 30.29	25.75 / 40.75 / 31.56	35.86 / 54.08 / 43.12
Baseline	60.88 / 63.78 / 62.30	44.99 / 31.78 / 37.25	29.07 / 39.52 / 33.50	42.62 / 47.84 / 45.08

Table 5: Comparison of Recall / Precision / F-score results on the test data set. Results on simple, binding, regulation, and all events are shown. GDep and Enju with PAS are used. Results by Miwa et al. (2010b), Björne et al. (2009), Riedel et al. (2009), and Baseline for Task 1 and Task 2 are shown for comparison. Baseline results are produced by removing dependency information from the parse results of GDep and Enju. The best score in each result is shown in bold.

tem.

6 Related Work

Many approaches for parser comparison have been proposed, and most comparisons have used gold treebanks with intermediate formats (Clegg and Shepherd, 2007; Pyysalo et al., 2007). Parser comparison has also been proposed on specific tasks such as unbounded dependencies (Rimell et al., 2009) and textual entailment (Önder Eker, 2009)[7]. Among them, application-oriented parser comparison across several formats was first introduced by Miyao et al. (2009), who compared eight parsers and five formats for the protein-protein interaction (PPI) extraction task. PPI extraction, the

[7] http://pete.yuret.com/

recognition of binary relations of between proteins, is one of the most basic information extraction tasks in the BioNLP field. Our findings do not conflict with those of Miyao et al. Event extraction can be viewed as an additional extrinsic evaluation task for syntactic parsers, providing more reliable and evaluation and a broader perspective into parser performance. An additional advantage of application-oriented evaluation on BioNLP shared task data is the availability of a manually annotated gold standard treebank, the GENIA treebank, that covers the same set of abstracts as the task data. This allows the gold treebank to be considered as an evaluation standard, in addition to comparison of performance in the primary task.

7 Conclusion

We compared six parsers and three formats on a bio-molecular event extraction task with a state-of-the-art event extraction system from two different aspects: dependency-based intrinsic evaluation and task-based extrinsic evaluation. The specific task considered was the BioNLP shared task, allowing the use of the GENIA treebank as a gold standard parse reference. Five of the six considered parsers were applied using biomedical models trained on the GENIA treebank, and they were found to produce similar performance. The comparison of the parsers from two aspects showed slightly different results, and and the dependency representations have advantages and disadvantages for the event extraction task.

The contributions of this paper are 1) the comparison of intrinsic and extrinsic evaluation on several commonly used parsers with a state-of-the-art system, and 2) demonstration of the limitation and possibility of the parser and system improvement on the task. One limitation of this study is that the comparison between the parsers is not perfect, as the parsers are used with the provided models, the format conversions miss some information from the original formats, and results with different formats depend on the ability of the event extraction system to take advantage of their strengths. To maximize comparability, the system was designed to extract features identically from similar parts of the dependency-based formats, further adding information provided by other formats, such as the lexical entries of the Enju format, from external resources. The results of this paper are expected to be useful as a guide not only for parser selection for biomedical information extraction but also for the development of event extraction systems.

The comparison in the present evaluation is limited to the dependency representation. As future work, it would be informative to extend the comparison to other syntactic representation, such as the PTB format. Finally, the evaluation showed that the system fails to recover approximately 40% of events even when provided with manually annotated treebank data, showing that other methods and resources need to be adopted to further improve bio-molecular event extraction systems. Such improvement is left as future work.

Acknowledgments

This work was partially supported by Grant-in-Aid for Specially Promoted Research (MEXT, Japan), Genome Network Project (MEXT, Japan), and Scientific Research (C) (General) (MEXT, Japan).

References

Bikel, Daniel M. 2004. A distributional analysis of a lexicalized statistical parsing model. In *In EMNLP*, pages 182–189.

Björne, Jari, Juho Heimonen, Filip Ginter, Antti Airola, Tapio Pahikkala, and Tapio Salakoski. 2009. Extracting complex biological events with rich graph-based feature sets. In *Proceedings of the BioNLP'09 Shared Task on Event Extraction*, pages 10–18.

Clegg, Andrew B. and Adrian J. Shepherd. 2007. Benchmarking natural-language parsers for biological applications using dependency graphs. *BMC Bioinformatics*, 8.

de Marneffe, Marie-Catherine and Christopher D. Manning. 2008. Stanford typed dependencies manual. Technical report, September.

de Marneffe, Marie-Catherine, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the IEEE / ACL 2006 Workshop on Spoken Language Technology*.

Johansson, Richard and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proceedings of NODALIDA 2007*, Tartu, Estonia, May 25-26.

Kim, Jin-Dong, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of bionlp'09 shared task on event extraction. In *BioNLP '09: Proceedings of the Workshop on BioNLP*, pages 1–9.

Klein, Dan and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 423–430, Morristown, NJ, USA. Association for Computational Linguistics.

McClosky, David. 2009. *Any Domain Parsing: Automatic Domain Adaptation for Natural Language Parsing*. Ph.D. thesis, Department of Computer Science, Brown University.

Miwa, Makoto, Sampo Pyysalo, Tadayoshi Hara, and Jun'ichi Tsujii. 2010a. A comparative study of syntactic parsers for event extraction. In *BioNLP2010: Proceedings of the Workshop on BioNLP*, Uppsala, Sweden, July.

Miwa, Makoto, Rune Sætre, Jin-Dong Kim, and Jun'ichi Tsujii. 2010b. Event extraction with complex event classification using rich features. *Journal of Bioinformatics and Computational Biology (JBCB)*, 8(1):131–146, February.

Miyao, Yusuke, Kenji Sagae, Rune Sætre, Takuya Matsuzaki, and Jun ichi Tsujii. 2009. Evaluating contributions of natural language parsers to protein-protein interaction extraction. *Bioinformatics*, 25(3):394–400.

Önder Eker. 2009. Parser evaluation using textual entailments. Master's thesis, Boğaziçi Üniversitesi, August.

Pyysalo, Sampo, Filip Ginter, Veronika Laippala, Katri Haverinen, Juho Heimonen, and Tapio Salakoski. 2007. On the unification of syntactic annotations under the stanford dependency scheme: A case study on bioinfer and genia. In *Biological, translational, and clinical language processing*, pages 25–32, Prague, Czech Republic, June. Association for Computational Linguistics.

Riedel, Sebastian, Hong-Woo Chun, Toshihisa Takagi, and Jun'ichi Tsujii. 2009. A markov logic approach to bio-molecular event extraction. In *BioNLP '09: Proceedings of the Workshop on BioNLP*, pages 41–49, Morristown, NJ, USA. Association for Computational Linguistics.

Rimell, Laura and Stephen Clark. 2009. Porting a lexicalized-grammar parser to the biomedical domain. *J. of Biomedical Informatics*, 42(5):852–865.

Rimell, Laura, Stephen Clark, and Mark Steedman. 2009. Unbounded dependency recovery for parser evaluation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 813–821, Singapore, August. Association for Computational Linguistics.

Sagae, Kenji and Jun'ichi Tsujii. 2007. Dependency parsing and domain adaptation with LR models and parser ensembles. In *EMNLP-CoNLL 2007*.

Tateisi, Yuka, Akane Yakushiji, Tomoko Ohta, and Junfichi Tsujii. 2005. Syntax annotation for the genia corpus. In *Proceedings of the IJCNLP 2005, Companion volume*, pages 222–227, Jeju Island, Korea, October.

Tsuruoka, Yoshimasa, Yuka Tateishi, Jin-Dong Kim, Tomoko Ohta, John McNaught, Sophia Ananiadou, and Jun'ichi Tsujii. 2005. Developing a robust part-of-speech tagger for biomedical text. In Bozanis, Panayiotis and Elias N. Houstis, editors, *Panhellenic Conference on Informatics*, volume 3746 of *Lecture Notes in Computer Science*, pages 382–392. Springer.

Entity-Focused Sentence Simplification for Relation Extraction

Makoto Miwa[1] **Rune Sætre**[1] **Yusuke Miyao**[2] **Jun'ichi Tsujii**[1,3,4]
[1]Department of Computer Science, The University of Tokyo
[2]National Institute of Informatics
[3]School of Computer Science, University of Manchester
[4]National Center for Text Mining
mmiwa@is.s.u-tokyo.ac.jp,rune.saetre@is.s.u-tokyo.ac.jp,
yusuke@nii.ac.jp,tsujii@is.s.u-tokyo.ac.jp

Abstract

Relations between entities in text have been widely researched in the natural language processing and information-extraction communities. The region connecting a pair of entities (in a parsed sentence) is often used to construct kernels or feature vectors that can recognize and extract interesting relations. Such regions are useful, but they can also incorporate unnecessary distracting information. In this paper, we propose a rule-based method to remove the information that is unnecessary for relation extraction. Protein–protein interaction (PPI) is used as an example relation extraction problem. A dozen simple rules are defined on output from a deep parser. Each rule specifically examines the entities in one target interaction pair. These simple rules were tested using several PPI corpora. The PPI extraction performance was improved on all the PPI corpora.

1 Introduction

Relation extraction (RE) is the task of finding a relevant semantic relation between two given target entities in a sentence (Sarawagi, 2008). Some example relation types are person–organization relations (Doddington et al., 2004), protein–protein interactions (PPI), and disease–gene associations (DGA) (Chun et al., 2006). Among the possible RE tasks, we chose the PPI extraction problem. PPI extraction is a major RE task; around 10 corpora have been published for training and evaluation of PPI extraction systems.

Recently, machine-learning methods, boosted by NLP techniques, have proved to be effective for RE. These methods are usually intended to highlight or select the relation-related regions in parsed sentences using feature vectors or kernels. The shortest paths between a pair of entities (Bunescu and Mooney, 2005) or pair-enclosed trees (Zhang et al., 2006) are widely used as focus regions. These regions are useful, but they can include unnecessary sub-paths such as appositions, which cause noisy features.

In this paper, we propose a method to remove information that is deemed unnecessary for RE. Instead of selecting the whole region between a target pair, the target sentence is simplified into simpler, pair-related, sentences using general, task-independent, rules. By addressing particularly the target entities, the rules do not affect important relation-related expressions between the target entities. We show how rules of two groups can be easily defined using the analytical capability of a deep parser with specific examination of the target entities. Rules of the first group can replace a sentence with a simpler sentence, still including the two target entities. The other group of rules can replace a large region (phrase) representing one target entity, with just a simple mention of that target entity. With only a dozen simple rules, we show that we can solve several simple well-known problems in RE, and that we can improve the performance of RE on all corpora in our PPI test-set.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 788–796,
Beijing, August 2010

2 Related Works

The general paths, such as the shortest path or pair-enclosed trees (Section 1), can only cover a part of the necessary information for relation extraction. Recent machine-learning methods specifically examine how to extract the missing information without adding too much noise. To find more representative regions, some information from outside the original regions must be included. Several tree kernels have been proposed to extract such regions from the parse structure (Zhang et al., 2006). Also the graph kernel method emphasizes internal paths without ignoring outside information (Airola et al., 2008). Composite kernels have been used to combine original information with outside information (Zhang et al., 2006; Miwa et al., 2009).

The approaches described above are useful, but they can include unnecessary information that distracts learning. Jonnalagadda and Gonzalez (2009) applied bioSimplify to relation extraction. BioSimplify is developed to improve their link grammar parser by simplifying the target sentence in a general manner, so their method might remove important information for a given target relation. For example, they might accidentally simplify a noun phrase that is needed to extract the relation. Still, they improved overall PPI extraction recall using such simplifications.

To remove unnecessary information from a sentence, some works have addressed sentence simplification by iteratively removing unnecessary phrases. Most of this work is not task-specific; it is intended to compress all information in a target sentence into a few words (Dorr et al., 2003; Vanderwende et al., 2007). Among them, Vickrey and Koller (2008) applied sentence simplification to semantic role labeling. With retaining all arguments of a verb, Vickrey simplified the sentence by removing some information outside of the verb and arguments.

3 Entity-Focused Sentence Simplification

We simplify a target sentence using simple rules applicable to the output of a deep parser called Mogura (Matsuzaki et al., 2007), to remove noisy information for relation extraction. Our method relies on the deep parser; the rules depend on the Head-driven Phrase Structure Grammar (HPSG) used by Mogura, and all the rules are written for the parser Enju XML output format. The deep parser can produce deep syntactic and semantic information, so we can define generally applicable comprehensive rules on HPSG with specific examination of the entities.

For sentence simplification in relation extraction, the meaning of the target sentence itself is less important than maintaining the truth-value of the relation (interact or not). For that purpose, we define rules of two groups: clause-selection rules and entity-phrase rules. A clause-selection rule constructs a simpler sentence (still including both target entities) by removing noisy information before and after the relevant clause. An entity-phrase rule simplifies an entity-containing region without changing the truth-value of the relation. By addressing the target entities particularly, we can define rules for many applications, and we can simplify target sentences with less danger of losing relation-related mentions. The rules are summarized in Table 1.

Our method is different from the sentence simplification in other systems (ref. Section 2). First, our method relies on the parser, while bioSimplify by Jonnalagadda and Gonzalez (2009) is developed for the improvement of their parser. Second, our method tries to keep only the relation-related regions, unlike other general systems including bioSimplify which tried to keep all information in a sentence. Third, our entity-phrase rules modify only the entity-containing phrases, while Vickrey and Koller (2008) tries to remove all information outside of the target verb and arguments.

3.1 Clause-selection Rules

In compound or complex sentences, it is natural to assume that one clause includes both the target entities and the relation-related information. It can also be assumed that the remaining sentence parts, outside the clause, contain less related (or noisy) information. The clause-selection rules simplify a sentence by retaining only the clause that includes the target entities (and by discarding the remainder of the sentence). We define three types of

Rule Group	Rule Type	#	Example (original → simplified)
Clause Selection	Sentence Clause	1	We show that A interacts with B. → A interacts with B.
	Relative Clause	2	... A that interacts with B. → A interacts with B.
	Copula	1	A is a protein that interacts with B. → A interacts with B.
Entity Phrase	Apposition	2	a protein, A → A
	Exemplification	4	proteins, such as A → A
	Parentheses	2	a protein (A) → A
	Coordination	3	protein and A → A

Table 1: Rules for Sentence Simplification. (# is the rule count. A and B are the target entities.)

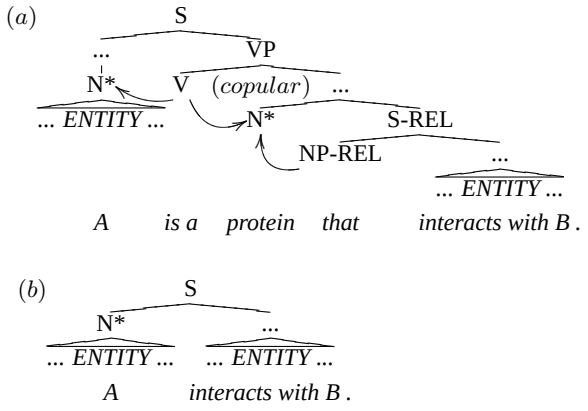


Figure 1: Copula Rule. (a) is simplified to (b). The arrows represent predicate–argument relations.

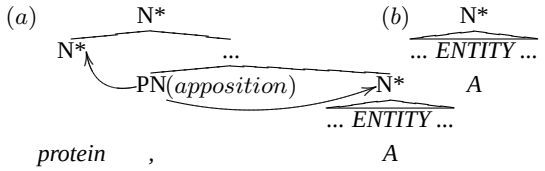


Figure 2: Apposition Rule.

clause-selection rules for sentence clauses, relative clauses, and copula. The *sentence clause rule* finds the (smallest) clause that includes both target entities. It then replaces the original sentence with the clause. The *relative clause rules* construct a simple sentence from a relative clause and the antecedent. If this simple sentence includes the target entities, it is used instead of the original sentence. We define two rules for the case where the antecedent is the subject of the relative clause. One rule is used when the relative clause includes both the target entities. The other rule is used when the antecedent includes one target entity and the relative clause includes the other target entity. The *copula rule* is for sentences that include copular verbs (e.g. be, is, become, etc). The rule constructs a simple sentence from a relative clause with the subject of the copular verb as the antecedent subject of the clause. The rule replaces the target sentence with the constructed sentence, if the relative clause includes one target entity and the subject of a copular verb includes the other target entity, as shown in Figure 1.

3.2 Entity-phrase Rules

Even the simple clauses (or paths between two target entities) include redundant or noisy expressions that can distract relation extraction. Some of these expressions are related to the target entities, but because they do not affect the truth-value of the relation, they can be deleted to make the path simple and clear. The target problem affects which expressions can be removed. We define four types of rules for appositions, exemplifications, parentheses, and coordinations. Two *apposition rules* are defined to select the correct element from an appositional expression. One element modifies or defines the other element in apposition, but the two elements represent the same information from the viewpoint of PPI. If the target entity is in one of these elements, removing the other element does not affect the truth-value of the interaction. Many of these apposition expressions are identified by the deep parser. The rule to select the last element is presented in Figure 2. Four *exemplification rules* are defined for the two major types of expressions using the phrases "including" or "such as". Exemplification is represented by hyponymy or hypernymy. As for appositions, the truth-value of the interaction does not change whether we use the specific mention or the hyperclass that the mention represents. Two *parentheses rules* are defined. Parentheses are useful for synonyms, hyponyms, or hypernyms (ref. the two

```
1: S ← input sentence
2: repeat
3:     reset rules {apply all the rules again}
4:     P ← parse S
5:     repeat
6:         r ← next rule {null if no more rules}
7:         if r is applicable to P then
8:             P ← apply r to P
9:             S ← sentence extracted from P
10:            break (Goto 3)
11:        end if
12:    until r is null
13: until r is null
14: return S
```

Figure 3: Pseudo-code for sentence simplification.

former rules). Three *coordination rules* are defined. Removing other phrases from coordinated expressions that include a target entity does not affect the truth-value of the target relation. Two rules are defined for simple coordination between two phrases (e.g. select left or right phrase), and one rule is defined to (recursively) remove one element from lists of more than two coordinated phrases (while maintaining the coordinating conjunction, e.g. "and").

3.3 Sentence Simplification

To simplify a sentence, we apply rules repeatedly until no more applications are possible as presented in Figure 3. After one application of one rule, the simplified sentence is re-parsed before attempting to apply all the rules again. This is because we require a consistent parse tree as a starting point for additional applications of the rules, and because a parser can produce more reliable output for a partly simplified sentence than for the original sentence. Using this method, we can also backtrack and seek out conversion errors by examining the cascade of partly simplified sentences.

4 Evaluation

To elucidate the effect of the sentence simplification, we applied the rules to five PPI corpora and evaluated the PPI extraction performance. We then analyzed the errors. The evaluation settings will be explained in Section 4.1. The results of the PPI extraction will be explained in Section 4.2. Finally, the deeper analysis results will be presented in Section 4.3.

4.1 Experimental Settings

The state-of-the-art PPI extraction system AkaneRE by Miwa et al. (2009) was used to evaluate our approach. The system uses a combination of three feature vectors: bag-of-words (BOW), shortest path (SP), and graph features. Classification models are trained with a support vector machine (SVM), and AkaneRE (with Mogura) is used with default parameter settings. The following two systems are used for a state-of-the-art comparison: AkaneRE with multiple parsers and corpora (Miwa et al., 2009), and Airola et al. (2008) single-parser, single-corpus system.

The rules were evaluated on the BioInfer (Pyysalo et al., 2007), AIMed (Bunescu et al., 2005), IEPA (Ding et al., 2002), HPRD50 (Fundel et al., 2006), and LLL (Nédellec, 2005) corpora[1]. Table 2 shows the number of positive (interacting) vs. all pairs. One duplicated abstract in the AIMed corpus was removed.

These corpora have several differences in their definition of entities and relations (Pyysalo et al., 2008). In fact, BioInfer and AIMed target all occurring entities related to the corpora (proteins, genes, etc). On the other hand, IEPA, HPRD50, and LLL only use limited named entities, based either on a list of entity names or on a named entity recognizer. Only BioInfer is annotated for other event types in addition to PPI, including static relations such as protein family membership. The sentence lengths are also different. The duplicated pair-containing sentences contain the following numbers of words on average: 35.8 in BioInfer, 31.3 in AIMed, 31.8 in IEPA, 26.5 in HPRD50, and 33.4 in LLL.

For BioInfer, AIMed, and IEPA, each corpus is split into training, development, and test datasets[2]. The training dataset from AIMed was the only training dataset used for validating the rules. The development datasets are used for error analysis. The evaluation was done on the test dataset, with models trained using training and development

[1] `http://mars.cs.utu.fi/PPICorpora/GraphKernel.html`

[2] This split method will be made public later.

	BioInfer		AIMed		IEPA		HPRD50		LLL	
	pos	all	pos	all	pos	all	pos	all	pos	all
training	1,848	7,108	684	4,072	256	630	-	-	-	-
development	256	928	102	608	23	51	-	-	-	-
test	425	1,618	194	1,095	56	136	-	-	-	-
all	2,534	9,653	980	5,775	335	817	163	433	164	330

Table 2: Number of positive (pos) vs. all possible sentence pairs in used PPI corpora.

	BioInfer			AIMed			IEPA		
Rule	Applied	F	AUC	Applied	F	AUC	Applied	F	AUC
No Application	0	62.5	83.0	0	61.2	87.9	0	73.4	82.5
Clause Selection	4,313	**63.5**	**83.9**	2,569	**62.5**	**88.2**	307	75.0	83.7
Entity Phrase	22,066	60.5	80.9	7,784	61.2	86.1	1,031	72.7	83.3
ALL	26,281	62.9	82.1	10,783	60.2	85.7	1,343	**75.4**	**85.7**

Table 3: Performance of PPI Extraction on test datasets. "Applied" represents the number of times the rules are applied on the corpus. "No Application" means PPI extraction without sentence simplification. ALL is the case all rules are used. The top scores for each corpus are shown in bold.

datasets). Ten-fold cross-validation (CV) was done to facilitate comparison with other existing systems. For HPRD50 and LLL, there are insufficient examples to split the data, so we use these corpora only for comparing the scores and statistics. We split the corpora for the CV, and measured the F-score (%) and area under the receiver operating characteristic (ROC) curve (AUC) as recommended in (Airola et al., 2008). We count each occurrence as one example because the correct interactions must be extracted for each occurrence if the same protein name occurs multiple times in a sentence.

In the experiments, the rules are applied in the following order: sentence–clause, exemplification, apposition, parentheses, coordination, copula, and relative-clause rules. Furthermore, if the same rule is applicable in different parts of the parse tree, then the rule is first applied closest to the leaf-nodes (deepest first). The order of the rules is arbitrary; changing it does not affect the results much. We conducted five experiments using the training and development dataset in IEPA, each time with a random shuffling of the order of the rules; the results were 77.8$\pm$0.26 in F-score and 85.9$\pm$0.55 in AUC.

4.2 Performance of PPI Extraction

The performance after rule application was better than the baseline (no application) on all the corpora, and most rules could be frequently applied. We show the PPI extraction performance on BioInfer, AIMed, and IEPA with rules of different groups in Table 3. The effect of using rules of different types for PPI extraction from HPRD50 and LLL is reported in Table 4 and Table 5. Table 6 shows the number of times each rule was applied in an "apply all-rules" experiment. The usability of the rules depends on the corpus, and different combinations of rules produce different

Rule	Applied	F	AUC
No Application	0	72.9	84.5
Sentence Clause	145	71.6	83.8
Relative Clause	7	73.3	84.1
Copula	0	72.9	84.5
Clause Selection	152	71.4	83.4
Apposition	64	73.2	84.6
Exemplification	33	72.9	84.7
Parentheses	90	72.9	85.1
Coordination	417	73.6	85.4
Entity Phrase	605	74.1	86.6
ALL	763	**75.0**	**86.6**

Table 4: Performance of PPI Extraction on HPRD50.

Rule	Applied	F	AUC
No Application	0	79.0	84.6
Sentence Clause	135	81.3	85.2
Relative Clause	42	78.8	84.6
Copula	0	79.0	84.6
Clause Selection	178	81.0	85.6
Apposition	197	79.6	83.9
Exemplification	0	79.0	84.6
Parentheses	56	79.5	85.8
Coordination	322	**84.2**	89.4
Entity Phrase	602	83.8	90.1
ALL	761	82.9	**90.5**

Table 5: Performance of PPI Extraction on LLL.

Rule	B	AIMed	IEPA	H	LLL
S. Cl.	3,960	2,346	300	150	135
R. Cl.	287	212	17	5	24
Copula	60	57	1	0	0
Cl. Sel.	4,307	2,615	318	155	159
Appos.	3,845	1,100	99	69	198
Exempl.	383	127	11	33	0
Paren.	2,721	2,158	235	91	88
Coord.	15,025	4,783	680	415	316
E. Foc.	21,974	8,168	1,025	608	602
Sum	26,281	10,783	1,343	763	761

Table 6: Distribution of the number of rules applied when all rules are applied. B:BioInfer, and H:HPRD50 corpora.

	Rules		Miwa et al.		Airola et al.	
	F	AUC	F	AUC	F	AUC
B	60.0	79.8	68.3	86.4	61.3	81.9
A	54.9	83.7	65.2	89.3	56.4	84.8
I	77.8	88.7	76.6	87.8	75.1	85.1
H	75.0	86.6	74.9	87.9	63.4	79.7
L	82.9	90.5	86.7	90.8	76.8	83.4

Table 7: Comparison with the results by Miwa et al. (2009) and Airola et al. (2008). The results with all rules are reported.

results. For the clause-selection rules, the performance was as good as or better than the baseline for all corpora, except for HPRD50, which indicates that the pair-containing clauses also include most of the important information for PPI extraction. Clause selection rules alone could improve the overall performance for the BioInfer and AIMed corpora. Entity-phrase rules greatly improved the performance on the IEPA, HPRD50, and LLL corpora, although these rules degraded the performance on the BioInfer and AIMed corpora. These phenomena hold even if we use small parts of the two corpora, so this is not because of the size of the corpora.

We compare our results with the results by Miwa et al. (2009) and Airola et al. (2008) in Table 7. On three of five corpora, our method provides better results than the state-of-the-art system by Airola et al. (2008), and also provides comparable results to those obtained using multiple parsers and corpora (Miwa et al., 2009) despite the fact that our method uses one parser and one corpus at a time. We cannot directly compare our result with Jonnalagadda and Gonzalez (2009) because the evaluation scheme, the baseline system, and test parts differ.

[FP→TN][Sentence, Parenthesis, Coordination] To characterize the AAV functions mediating this effect, cloned AAV type 2 wild-type or mutant genomes were transfected into simian virus 40 (SV40)-transformed hamster cells together with the six HSV replication genes (encoding UL5, UL8, major DNA-binding protein, **DNA polymerase**, UL42 , and **UL52**) which together are necessary and sufficient for the induction of SV40 DNA amplification (R. Heilbronn and H. zur Hausen, J. Virol. 63:3683-3692, 1989). (BioInfer.d760.s0)
[TP→FN][Coordination] Both the **GT155-calnexin** and the GT155-CAP-60 interactions were dependent on the presence of a correctly modified oligosaccharide group on GT155, a characteristic of many calnexin interactions. (AIMed.d167.s1408)
[TN→TN][Coordination, Parenthesis] **Leptin** may act as a negative feedback signal to the hypothalamic control of appetite through suppression of **neuropeptide Y** (NPY) secretion and stimulation of cocaine and amphetamine regulated transcript (CART) . (IEPA.d190.s454)

Figure 4: A rule-related error, a critical error, and a parser-related error. Regions removed by the rules are underlined, and target proteins are shown in bold. Predictions, applied rules, and sentence IDs are shown.

[FN→TP][Sentence, Coordination] **WASp** contains a binding motif for the Rho GTPase CDC42Hs as well as **verprolin** / cofilin-like actin-regulatory domains , but no specific actin structure regulated by CDC42Hs-WASp has been identified. (BioInfer.d795.s0)
[FN→TP][Parenthesis, Apposition] The protein **Raf-1** , a key mediator of mitogenesis and differentiation, associates with **p21ras** (refs 1-3) . (AIMed.d124.s1055)
[FN→TP][Sentence, Parenthesis] On the basis of far-Western blot and plasmon resonance (BIAcore) experiments, we show here that recombinant **bovine prion protein** (bPrP) (25-242) strongly interacts with the catalytic alpha/alpha' subunits of **protein kinase CK2** (also termed 'casein kinase 2') (IEPA.d197.s479)

Figure 5: Correctly simplified cases. The first sentence is a difficult (not PPI) relation, which is typed as "Similar" in the BioInfer corpus.

4.3 Analysis

We trained models using the training datasets and classified the examples in the development datasets. Two types of analysis were performed based on these results: *simplification-based* and *classification-based analysis*.

For the *simplification-based analysis*, we compared positive (interacting) and negative pair sentences that produce the exact same (inconsistent) sentence after protein names normalization and

	BioInfer				AIMed				IEPA				
Before simplification After simplification	FN TP	FP TN	TP FN	TN FP	FN TP	FP TN	TP FN	TN FP	FN TP	FP TN	TP FN	TN FP	Not Affected
No Error	18	2	3	35	14	21	21	8	3	2	0	4	32
No Application	3	2	0	3	0	7	8	0	0	1	0	1	7
Number of Errors	0	2	0	32	4	2	1	4	0	0	0	0	1
Number of Pairs	21	6	3	70	18	30	30	12	3	3	0	5	40
Coordination	0	0	0	20	4	2	1	0	0	0	0	0	1
Sentence	0	2	0	4	0	0	0	4	0	0	0	0	0
Parenthesis	0	0	0	5	0	0	0	0	0	0	0	0	0
Exemplification	0	0	0	2	0	0	0	0	0	0	0	0	0
Apposition	0	0	0	1	0	0	0	0	0	0	0	0	0

Table 8: Distribution of sentence simplification errors compared to unsimplified predictions with their types (on the three development datasets). TP, True Positive; TN, True Negative; FN, False Negative; FP, False Positive. "No Error" means that simplification was correct; "No Application" means that no rule could be applied; Other rule names mean that an error resulted from that rule application. "Not Affected" means that the prediction outcome did not change.

simplification in the training dataset. The numbers of such inconsistent sentences are 7 for BioInfer, 78 for AIMed, and 1 for IEPA. The few inconsistencies in BioInfer and IEPA are from errors by the rules, mainly triggered by parse errors. The frequent inconsistencies in AIMed are mostly from inconsistent annotations. For example, even if all coordinated proteins are either interacting or not, only the first protein mention is annotated as interacting.

For the *classification-based analysis*, we specifically examine simplified pairs that were predicted differently before and after the simplification. Pairs predicted differently before and after rule application were selected: 100 random pairs from BioInfer and all 90 pairs from AIMed. For IEPA, all 51 pairs are reported. Simplified results are classified as errors when the rules affect a region unrelated to the entities in the smallest sentence clause. The results of analysis are shown in Table 8. There were 34 errors in BioInfer, and 11 errors in AIMed. Among the errors, there were five *critical errors* (in two sentences, in AIMed). Critical errors mean that the pairs lost relation-related mentions, and the errors are the only errors which caused the changes in the truth-value of the relation. There was also a *rule-related error* (in BioInfer), which means that rules with correct parse results affect a region unrelated to the entities, and parse errors (*parser-related errors*). Figure 4 shows the rule-related error in BioInfer, one critical error in AIMed, and one parser-related error in IEPA.

5 Discussion

Our end goal is to provide consistent relation extraction for real tasks. Here we discuss the "safety" of applying our simplification rules, the difficulties in the BioInfer and AIMed corpora, the reduction of errors, and the requirements for such a general (PPI) extraction system.

Our rules are applicable to sentences, with little danger of changing the relation-related mentions. Figure 5 shows three successfully simplified cases ("No Error" cases from Table 8). The sentence simplification leaves sufficient information to determine the value of the relation in these examples. Relation-related mentions remained for most of the simplification error cases. There were only five critical errors, which changed the truth-value of the relation, out of 46 errors in 241 pairs shown in Table 8. Please note that some rules can be dangerous for other relation extraction tasks. For example, the *sentence clause rule* could remove modality information (negation, speculation, etc.) modifying the clause, but there are few such cases in the PPI corpora (see Table 8). Also, the task of hedge detection (Morante and Daelemans, 2009) can be solved separately, in the original sentences, after the interacting pairs have been found. For example, in the BioNLP shared task challenge and the BioInfer corpus, interaction detection and modality are treated as two different tasks. Once other NLP tasks, like static relation (Pyysalo et

al., 2009) or coreference resolution, become good enough, they can supplement or even substitute some of the proposed rules.

There are different difficulties in the BioInfer and AIMed corpora. BioInfer includes more complicated sentences and problems than the other corpora do, because 1) the apposition, coordination, and exemplification rules are more frequently used in the BioInfer corpus than in the other corpora (shown in Table 6), 2) there were more errors in the BioInfer corpus than in other corpora among the simplified sentences (shown in Table 8), and 3) BioInfer has more words per sentence and more relation types than the other corpora. AIMed contains several annotation inconsistencies as explained in Section 4.3. These inconsistencies must be removed to properly evaluate the effect of our method.

Simplification errors are mostly caused by parse errors. Our rule specifically examines a part of parser output; a probability is attached to the part. The probability is useful for defining the order of rule applications, and the n-best results by the parser are useful to fix major errors such as coordination errors. By using these modifications of rule applications and by continuous improvement in parsing technology for the biomedical domain, the performance on the BioInfer and AIMed corpora will be improved also for the all rules case.

The PPI extraction system lost the ability to capture some of the relation-related expressions left by the simplification rules. This indicates that the system used to extract some relations (before simplification) by using back-off features like bag-of-words. The system can reduce bad effects caused by parse errors, but it also captures the annotation inconsistencies in AIMed. Our simplification (without errors) can capture more general expressions needed for relation extraction. To provide consistent PPI relation extraction in a general setting (e.g. for multiple corpora or for other public text collections), the parse errors must be dealt with, and a relation extraction system that can capture (only) general relation-related expressions is needed.

6 Conclusion

We proposed a method to simplify sentences, particularly addressing the target entities for relation extraction. Using a few simple rules applicable to the output of a deep parser called Mogura, we showed that sentence simplification is effective for relation extraction. Applying all the rules improved the performance on three of the five corpora, while applying only the clause-selection rules raised the performance for the remaining two corpora as well. We analyzed the simplification results, and showed that the simple rules are applicable with little danger of changing the truth-values of the interactions.

The main contributions of this paper are: 1) explanation of general sentence simplification rules using HPSG for relation extraction, 2) presenting evidence that application of the rules improve relation extraction performance, and 3) presentation of an error analysis from two viewpoints: simplification and classification results.

As future work, we are planning to refine and complete the current set of rules, and to cover the shortcomings of the deep parser. Using these rules, we can then make better use of the parser's capabilities. We will also attempt to apply our simplification rules to other relation extraction problems than those of PPI.

Acknowledgments

This work was partially supported by Grant-in-Aid for Specially Promoted Research (MEXT, Japan), Genome Network Project (MEXT, Japan), and Scientific Research (C) (General) (MEXT, Japan).

References

Airola, Antti, Sampo Pyysalo, Jari Björne, Tapio Pahikkala, Filip Ginter, and Tapio Salakoski. 2008. A graph kernel for protein-protein interaction extraction. In *Proceedings of the BioNLP 2008 workshop*.

Bunescu, Razvan C. and Raymond J. Mooney. 2005. A shortest path dependency kernel for relation extraction. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 724–731.

Bunescu, Razvan C., Ruifang Ge, Rohit J. Kate, Edward M. Marcotte, Raymond J. Mooney, Arun K. Ramani, and Yuk Wah Wong. 2005. Comparative experiments on learning information extractors for proteins and their interactions. *Artificial Intelligence in Medicine*, 33(2):139–155.

Chun, Hong-Woo, Yoshimasa Tsuruoka, Jin-Dong Kim, Rie Shiba, Naoki Nagata, Teruyoshi Hishiki, and Jun'ichi Tsujii. 2006. Extraction of gene-disease relations from medline using domain dictionaries and machine learning. In *The Pacific Symposium on Biocomputing (PSB)*, pages 4–15.

Ding, J., D. Berleant, D. Nettleton, and E. Wurtele. 2002. Mining medline: abstracts, sentences, or phrases? *Pacific Symposium on Biocomputing*, pages 326–337.

Doddington, George, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ACE) program: Tasks, data, and evaluation. In *Proceedings of LREC'04*, pages 837–840.

Dorr, Bonnie, David Zajic, and Richard Schwartz. 2003. Hedge trimmer: A parse-and-trim approach to headline generation. In *in Proceedings of Workshop on Automatic Summarization*, pages 1–8.

Fundel, Katrin, Robert Küffner, and Ralf Zimmer. 2006. Relex—relation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371.

Jonnalagadda, Siddhartha and Graciela Gonzalez. 2009. Sentence simplification aids protein-protein interaction extraction. In *Proceedings of the 3rd International Symposium on Languages in Biology and Medicine*, pages 109–114, November.

Matsuzaki, Takuya, Yusuke Miyao, and Jun'ichi Tsujii. 2007. Efficient HPSG parsing with supertagging and cfg-filtering. In *IJCAI'07: Proceedings of the 20th international joint conference on Artifical intelligence*, pages 1671–1676, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Miwa, Makoto, Rune Sætre, Yusuke Miyao, and Jun'ichi Tsujii. 2009. Protein-protein interaction extraction by leveraging multiple kernels and parsers. *International Journal of Medical Informatics*, June.

Morante, Roser and Walter Daelemans. 2009. Learning the scope of hedge cues in biomedical texts. In *Proceedings of the BioNLP 2009 Workshop*, pages 28–36, Boulder, Colorado, June. Association for Computational Linguistics.

Nédellec, Claire. 2005. Learning language in logic - genic interaction extraction challenge. In *Proceedings of the LLL'05 Workshop*.

Pyysalo, Sampo, Filip Ginter, Juho Heimonen, Jari Björne, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. 2007. BioInfer: A corpus for information extraction in the biomedical domain. *BMC Bioinformatics*, 8:50.

Pyysalo, Sampo, Antti Airola, Juho Heimonen, Jari Björne, Filip Ginter, and Tapio Salakoski. 2008. Comparative analysis of five protein-protein interaction corpora. In *BMC Bioinformatics*, volume 9(Suppl 3), page S6.

Pyysalo, Sampo, Tomoko Ohta, Jin-Dong Kim, and Jun'ichi Tsujii. 2009. Static relations: a piece in the biomedical information extraction puzzle. In *BioNLP '09: Proceedings of the Workshop on BioNLP*, pages 1–9, Morristown, NJ, USA. Association for Computational Linguistics.

Sarawagi, Sunita. 2008. Information extraction. *Foundations and Trends in Databases*, 1(3):261–377.

Vanderwende, Lucy, Hisami Suzuki, Chris Brockett, and Ani Nenkova. 2007. Beyond sumbasic: Task-focused summarization with sentence simplification and lexical expansion. *Inf. Process. Manage.*, 43(6):1606–1618.

Vickrey, David and Daphne Koller. 2008. Sentence simplification for semantic role labeling. In *Proceedings of ACL-08: HLT*, pages 344–352, Columbus, Ohio, June. Association for Computational Linguistics.

Zhang, Min, Jie Zhang, Jian Su, and Guodong Zhou. 2006. A composite kernel to extract relations between entities with both flat and structured features. In *ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 825–832. Association for Computational Linguistics.

Using Cross-Lingual Projections to Generate Semantic Role Labeled Corpus for Urdu - A Resource Poor Language

Smruthi Mukund
CEDAR
University at Buffalo
smukund@buffalo.edu

Debanjan Ghosh
Thomson Reuters R&D
debanjan.ghosh@
thomsonreuters.com

Rohini K. Srihari
CEDAR
University at Buffalo
rohini@cedar.buffalo.edu

Abstract

In this paper we explore the possibility of using cross lingual projections that help to automatically induce role-semantic annotations in the PropBank paradigm for Urdu, a resource poor language. This technique provides annotation projections based on word alignments. It is relatively inexpensive and has the potential to reduce human effort involved in creating semantic role resources. The projection model exploits lexical as well as syntactic information on an English-Urdu parallel corpus. We show that our method generates reasonably good annotations with an accuracy of 92% on short structured sentences. Using the automatically generated annotated corpus, we conduct preliminary experiments to create a semantic role labeler for Urdu. The results of the labeler though modest, are promising and indicate the potential of our technique to generate large scale annotations for Urdu.

1 Introduction

Semantic Roles (also known as thematic roles) help to understand the semantic structure of a document (Fillmore, 1968). At a fundamental level, they help to capture the similarities and differences in the meaning of verbs via the arguments they define by generalizing over surface syntactic configurations. In turn, these roles aid in domain independent understanding as the semantic frames and semantic understanding systems do not depend on the syntactic configuration for each new application domain. Identifying semantic roles benefit several language processing tasks - information extraction (Surdeanu *et al.*, 2003), text categorization (Moschitti, 2008) and finding relations in textual entailment (Burchardt and Frank 2006).

Automatically identifying semantic roles is often referred to as shallow semantic parsing (Gildea and Jurafsky, 2002). For English, this process is facilitated by the existence of two main SRL annotated corpora – FrameNet (Baker *et al.*, 1998) and PropBank (Palmer *et al.*, 2005). Both datasets mark almost all surface realizations of semantic roles. FrameNet has 800 semantic frames that cover 120,000 example sentences[1]. PropBank has annotations that cover over 113,000 predicate-argument structures. Clearly English is well supported with resources for semantic roles. However, there are other widely spoken resource poor languages that are not as privileged. The PropBank based resources available for languages like Chinese (Xue and Palmer, 2009), Korean (Palmer *et al.*, 2006) and Spanish (Taule, 2008) are only about two-thirds the size of the English PropBank.

Several alternative techniques have been explored in the literature to generate semantic role labeled corpora for resource poor languages as providing manually annotated data is time consuming and involves intense human labor. Ambati and Chen (2007) have conducted an extensive survey and outlined the benefits of using parallel corpora to transfer annotations. A wide range of annotations from part of speech (Hi and Hwa, 2005) and chunks (Yarowsky *et al.*, 2001) to word senses (Diab and Resnik, 2002), dependencies (Hwa *et al.*, 2002) and semantic roles (Pado and Lapata, 2009) have been successfully transferred between languages. FrameNet style annotations in Chinese is obtained by mapping English FrameNet entries directly to concepts listed in HowNet[2] (online ontology for Chinese) with an accuracy of 68% (Fung and Chen, 2004).

[1] Wikipedia - http://en.wikipedia.org/wiki/PropBank
[2] http://www.keenage.com/html/e_index.html

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 797–805,
Beijing, August 2010

Fung *et al.* (2007) analyze an automatically annotated English-Chinese parallel corpus and show high cross-lingual agreement for PropBank roles (range of 75%-95% based on the roles).

In this paper we explore the possibility of using English-Urdu parallel corpora to generate SRL annotations for Urdu, a less commonly taught language (LCTL). Earlier attempts to generate SRL corpora using annotation projections have been for languages such as German, French (Pado and Lapata, 2009) and Italian (Moschitti, 2009) that have high vocabulary overlap with English. Also, German belongs to the same language family as English (Germanic family). Urdu on the other hand is an Indic language that is grammatically very different and shares almost no vocabulary with English.

The technique of cross lingual projections warrants good BLEU score that ensures correct word alignments. According to NIST 2008 Open Machine Translation challenge [3], a 0.2280 best BLEU score was achieved for Urdu to English translation. This is comparable to the BLEU scores achieved for German to English – 0.253 and French to English – 0.3 (Koehn, 2005). But, for SRL transfer, perfect word alignment is not mandatory as SRL requires semantic correspondence only. According to Fillmore (1982) semantic frames are based on conceptual structures. They are generalizations over surface structures and hence less prone to syntactic variations. Since English and Urdu have a reasonable semantic correspondence (Example 3), we believe that the projections when capped with a post processing step will considerably reduce the noise induced by inaccurate alignments and produce acceptable mappings.

Hindi is syntactically similar to Urdu. These languages are standardized forms of Hindustani. They are free word order languages and follow a general SOV (Subject-Object-Verb) structure. Projection approach has been used by (Mukerjee *et al.*, 2006) and (Sinha, 2009) to transfer verb predicates from English onto Hindi. Sinha (2009) achieves a 90% F-Measure in verb predicate transfer from English to Hindi. This shows that using cross lingual transfer approach to obtain semantic annotations for Urdu from English is an idea worth exploring.

[3]http://www.itl.nist.gov/iaui/894.01/tests/mt/2008/doc/mt08_official_results_v0.html

1.1 Approach

Our approach leverages existing English PropBank annotations provided via the SemLink[4] corpus. SemLink provides annotations for VerbNet using the **pb (P**rop**B**ank) attribute. By using English-Urdu parallel corpus we acquire verb predicates and their arguments. When we transfer verb predicates (lemmas), we also transfer **pb** attributes. We obtain annotation projections from the parallel corpora as follows:

1. Take a pair of sentences *E* (in English) and *U* (in Urdu) that are translations of each other.
2. Annotate *E* with semantic roles.
3. Project the annotations from *E* onto *U* using word alignment information, lexical information and linguistic rules that involve syntactic information.

There are several challenges to the annotation projection technique. Dorr (1994) presents some major lexical-semantic divergence problems applicable in this scenario:

(a) Thematic Divergence - In some cases, although there exists semantic parallelism, the theme of the English sentence captured in the subject changes into an object in the Urdu sentence (Example 1).

(b) Conflatational Divergence - Sometimes target translations spans over a group of words (Example 1: *plays* is mapped to *kirdar ada)*. Trying to ascertain this word span for semantic roles is difficult as the alignments can be incomplete and very noisy.

(c) Demotional divergence and Structural divergence - Despite semantic relatedness, in some sentence pairs, alignments obtained from simple projections generate random matchings as the usage is syntactically dissimilar (Example 2).

Handling all challenges adds complexity to our model. The heuristic rules that we implement are guided by linguistic knowledge of Urdu. This increases the effectiveness of the alignments.

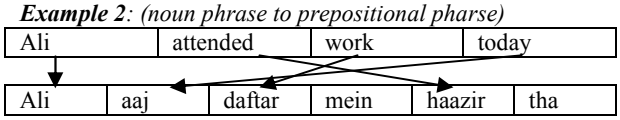
Example 1:

I *(subject)*	am	Angry	at	Reheem *(object)*
Raheem *(subject)*	mujhe *(object)*	Gussa	dilate	hai

(Raheem brings anger in me)

[4] http://verbs.colorado.edu/semlink/

***Example 2**: (noun phrase to prepositional pharse)*



(Ali was present at work today)

2 Generating Parallel Corpora

PropBank provides SRL annotated corpora for English. It uses predicate independent labels (ARG0, ARG1, etc.) which indicate how a verb relates to its arguments. The argument types are consistent across all uses of a single verb and do not consider the sense of the verb. We use the PropBank annotations provided for the Wall Street Journal (WSJ) part of the Penn Tree bank corpus (Marcus *et al.*, 2004). The arguments of a verb are labeled sequentially from ARG0 to ARG5 where ARG0 is the proto-typical Agent, ARG1 is the proto-typical patient, ARG2 is the recipient, and so on. There are other adjunct tags in the dataset that are indicated by ARGM that include tags for location (ARGM-LOC), temporal tags (ARGM-TMP) etc.

An Urdu corpus of 6000 sentences corresponding to 317 WSJ articles of Penn Tree Bank corpus is provided by CRULP[5] (used in the NIST 2008 machine translation task). We consider 2350 English sentences with PropBank annotations that have corresponding Urdu translations (CRULP corpus) for our experiments.

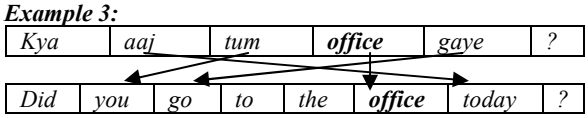
2.1 Sentence Alignment

Sentence alignment is a prerequisite for any parallel corpora processing. As the first step, we had to generate a perfect sentence aligned parallel corpus as the translated sentences, despite belonging to the same domain (WSJ – Penn tree bank), had several errors in demarcating the sentence boundaries.

Sentence alignment between English and Urdu is achieved over two iterations. In the first iteration, the length of each sentence is calculated based on the occurrence of words belonging to important part of speech categories such as proper nouns, adjectives and verbs. Considering main POS categories for length assessment helps overcome the conflatational divergence issue. For each English sentence, Urdu sentences with the same length are considered to be probable candidates for alignment. In the second iteration, an Urdu-English lexicon is used on the Urdu corpus and English translations are obtained. An English-Urdu sentence pair with maximum lexical match is considered to be sentence aligned.

Clearly this method is highly dependent on the existence of an exhaustive Urdu-English dictionary. The lexicons that we use to perform lookups are collected by mining Wikipedia and other online resources (Mukund *et al.,* 2010). However, lexicon lookups will fail for Out-Of-Vocabulary words. There could also be a collision if Urdu sentences have English transliterated words (Example 3, *"office"*). Such errors are manually verified for correctness.

Example 3:



2.2 Word Alignment

In the case of generating word alignments it is beneficial to calculate alignments in both translation directions (English – Urdu and Urdu - English). This nature of symmetry will help to reduce alignment errors. We use the Berkeley Aligner[6] word alignment package which implements a joint training model with posterior decoding (Liang *et al.*, 2006) to consider bidirectional alignments. Predictions are made based on the agreements obtained by two bidirectional models in the training phase. The intuitive objective function that incorporates data likelihood and a measure of agreement between the models is maximized using an EM-like algorithm. This alignment model is known to provide 29% reduction in AER over IBM model 4 predictions.

On our data set the word alignment accuracy is 71.3% (calculated over 200 sentence pairs). In order to augment the alignment accuracy, we added 3000 Urdu-English words and phrases obtained from the Urdu-English dictionary to our parallel corpus. The alignment accuracy improved by 3% as the lexicon affects the word co-occurrence count.

Word alignment in itself does not produce accurate semantic role projections from English to Urdu. This is because the verb predicates in Urdu can span more than one token. Semantic roles

[5]http://www.crulp.org/

[6] http://nlp.cs.berkeley.edu/Main.html

can cover sentential constituents of arbitrary length, and simply using word alignments for projection is likely to result in wrong role spans. Also, alignments are not obtained for all words. This could lead to missing projections.

One way to correct these alignment errors is to devise token based heuristic rules. This is not very beneficial as writing generic rules is difficult and different errors demand specific rules. We propose a method that considers POS, tense and chunk information along with word alignments to project annotations.

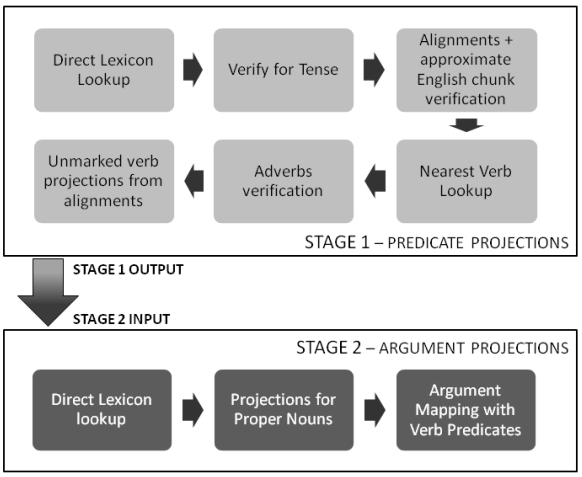


Figure 1: Projection model

Our proposed approach can be explained in two stages as shown in figure 1. In Stage 1 only verb predicates are transferred from English to Urdu. Stage 2 involves transfer of arguments and depends on the output of Stage 1. Predicate transfer cannot rely entirely on word alignments (§3). Rules devised around the chunk boundaries boost the verb predicate recognition rate.

Any verb group sequence consisting of a main verb and its auxiliaries are marked as a verb chunk. Urdu data is tagged using the chunk tag set proposed exclusively for Indian languages by Bharati *et al.*, (2006). Table 1 shows the tags that are important for this task.

Verb Chunk	Description
VGF	Verb group is finite (decided by the auxiliaries)
VGNF	Verb group for non-finite adverbial and adjectival chunk
VGNN	Verb group has a gerund

Table 1: Verb chunk tags in Urdu

The sentence aligned parallel corpora that we feed as input to our model is POS tagged for both English and Urdu. Urdu data is also tagged for chunk boundaries and morphological features like tense, gender and number information. Named Entities are also marked on the Urdu data set as they help in tagging the ARGM arguments. All the NLP taggers (POS, NE, Chunker, and Morphological Analyzer) used in this work are detailed in Mukund *et al.,* (2010).

English data is not chunked using a conventional chunk tagger. Each English sentence is split into virtual phrases at boundaries determined by the following parts of speech – IN, TO, MD, POS, CC, DT, SYM,: (Penn Tree Bank tagset). These tags represent positions in a sentence that typically mark context transitions (they are mostly the closed class words). We show later how these approximate chunks assist in correcting predicate mappings.

We use an Urdu-English dictionary (§2.1) that assigns English meanings to Urdu words in each sentence. Using translation information from a dictionary can help transfer verb predicates when the translation equivalent preserves the lexical meaning of the source language.

The first rule that gets applied for predicate transfer is based on lexicon lookup. If the English verb is found to be a synonym to an Urdu word that is part of a verb chunk, then the lemma associated with the English word is transferred to the entire verb chunk in Urdu. However not all translations' equivalents are lexically synonymous. Sometimes the word used in Urdu is different in meaning to that in English but relevant in the context (lexical divergence).

The word alignments considered in proximity to the approximate English chunks come to rescue in such scenarios. Here, for all the words occurring in each Urdu verb chunk, corresponding English aligned words are found from the word alignments. If the words that are found belong to the same approximate English chunk, then the verb predicate of that chunk (if present) is projected onto the verb chunk in Urdu. This heuristic technique increases the verb projection accuracy by about 15% as shown in §4.

The Penn tree bank tag set for English part of speech has different tags for verbs based on the tense information. VBD is used to indicate past tense, and VBP and VBZ for present tense. Urdu

also has the tense information associated with the verbs in some cases. We exploit this similarity to project the verb predicates from English onto Urdu.

The adverbial chunk in Urdu includes pure adverbial phrases. These chunks also form part of the verb predicates.

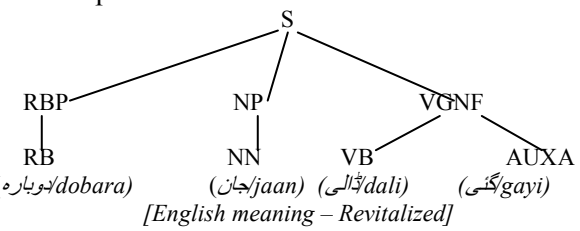


Figure 2: example for demotional divergence

E.g. consider the English word "revitalized" (figure 2). This is tagged VBD. However, the Urdu equivalent of this word is *"دوبارہ جان ڈالی گئی" (dobara jaan daali gayi ~ to put life in again)*. The POS tags are "RB, NN, VB, AUXA" *(adverb, noun, verb, aspectual auxiliary)*. The word *"dobara"* is a part of the adverbial chunk RBP and the infinite verb chunk VGNF spans across the last two words *"daali gayi"*. *"jaan"* is a noun chunk. This kind of demotional divergence is commonly observed in languages like Hindi and Urdu. In order to consider this entire phrase to be the Urdu equivalent representation of the English word *"revitalized"*, a rule for adverbial chunk is included as the last step to account for unaccommodated English verbs in the projections.

In the PropBank corpus, predicate argument relations are marked for almost all occurrences of non-copula verbs. We however do not have POS tags that help to identify non-copula words. Words that can be auxiliary verbs occur as non-copula verbs in Urdu. We maintain a list of such auxiliary verbs. When the verb chunk in Urdu contains only one word and belongs to the list, we simply ignore the verb chunk and proceed to the next chunk. This avoids several false positives in verb projections.

Stage 2 of the model includes the transfer of arguments. In order to see how well our method works, we project all argument annotations from English onto Urdu. We do not consider word alignments for arguments with proper nouns. The double metaphone algorithm (Philips 2000) is applied on both English NNP (proper noun) tagged words as well as English transliterated Urdu (NNP) tagged words. Arguments from English are mapped onto Urdu for word pairs with the same metaphone code.

For other arguments, we consider word alignments in proximity to verb predicates. The argument boundaries are determined based on chunk and POS information. We observe that our method projects the annotations associated with nouns fairly well. However, when the arguments contain adjectives, the boundaries are discontinuous. In such cases, we consider the entire chunk without the case marker as a probable candidate for the projected argument. We also have some issues with the ARGM-MOD arguments in that they overlap with the verb predicates. When the verb predicate that it overlaps with is a complex predicate, we consider the entire verb chunk to be the Urdu equivalent candidate argument. These rules along with word alignments yield fairly accurate projections.

The rules that we propose are dependent on the POS, chunk and tense information that are language specific. Hence our method is language independent only to the extent that the new language considered should have similar syntactic structure as Urdu. Indic languages fall in this category.

3 Verb Predicates

Detecting verb predicates can be a challenging task especially if very reliable and efficient tools such as POS tagger and chunkers are not available. We apply the POS tagger (CRULP tagset, 88% F-Score) and Chunker (Hindi tagset, 90% F-Score) provided by Mukund *et al.,* (2010) on the Urdu data set and show that syntactic information helps to compensate alignment errors. Stanford POS tagger[7] (Penn Tree bank tagset) is applied on the English data set.

Predicates can be simple predicates that lie within the chunk boundary or complex predicates when they span across chunk boundaries. When verbs in English are expressed in Urdu/Hindi, in several cases, more than one word is used to achieve perfect translation. In English the tense of the verb is mostly captured by the verb morpheme such as *"ask**ed**" "sa**id**" "say**ing**"*. In Urdu the tense is mostly captured by the auxiliary verbs. So a single word English verb such as *"talking"* would be translated into two words

[7] http://nlp.stanford.edu/software/tagger.shtml

"batein karna" where *"karna"~ do* is the auxiliary verb. However this cannot be generalized as there are instances when translations are word to word. E.g. *"said"* is mapped to a single word Urdu verb *"kaha"*.

Complex predicates in Urdu can occur in the following POS combinations. *Noun+Verb, Adjective+Verb, Verb+Verb, Adverb+Verb*. Table 2 lists the main verb tags present in the Urdu POS tagset. (refer Penn Tree bank POS tagset for English tags).

Urdu Tags	Description
VB	Verb
VBI	Infinitive Verb
VBL	Light Verb
VBLI	Infinitive Light Verb
VBT	Verb to be
AUXA	Aspectual Auxiliary
AUXT	Tense Auxiliary

Table 2: Verb tags

Auxiliary verbs in Urdu occur alongside VB, VBI, VBL or VBLI tags. Sinha (2009) defines complex predicates as a group of words consisting of a noun (NN/NNP), an adjective (JJ), a verb (VB) or an adverb (RB) followed by a light verb (VBL/VBLI). Light verbs are those which contribute to the tense and agreement of the verb (Butt and Geuder, 2001). However, despite the existence of a light verb tag, it is noticed that in several sentences, verbs followed by auxiliary verbs need to be grouped as a single predicate. Hence, we consider such combinations as belonging to the complex predicate category.

ENG- *According*_VBG *to*_TO *some*_DT *estimates*_NNS *the*_DT *rule*_NN *changes*_NNS *would*_MD ***cut_VB*** *insider*_NN *filings*_NNS *by*_IN *more*_JJR *than*_IN *a*_DT *third*_JJ
URD- [*Kuch*_QN *andaazon*_NN *ke*_CM *mutabiq*_NNCM]_NP [*kanoon*_NN *mein*_CM]_NP [*tabdeeliayan*_NN]_NP[*androni*_JJ *drjbndywn*_NN *ko*_CM]_NP [*ayk*_CD *thayiy*_FR *se*_CM]_NP [***zyada***_I ***kam***_JJ]_JJP [***karey***_VBL ***gi***_AUXT]_VGF

Example 4

Example 4 demonstrates the existence of a light verb in a complex predicate. The English verb *"cut"* is mapped to *"کم کریں گی" (kam karey gi)* belonging to the VBF chunk group.

ENG- *Rolls*_NNP *-_: Royce*_NNP *Motor*_NNP *Cars*_NNPS *Inc.*_NNP ***said***_VBD *it*_PRP ***expects***_VBZ *its*_PRP*$ U.S.*_NNP *sales*_NNS *to*_TO ***remain***_VB *steady*_JJ *at*_IN *about*_IN *1 200*_CD *cars*_NNS *in*_IN *1990*_CD

URD - [*Rolls Royce motor car inc*_NNPC *ne*_CM]_NP [***kaha*_VB**]_VBNF [*wo*_PRP]_NP [*apney*_PRRFP$]_NP [*U.S.*_NNP *ki*_CM]_NP [*frwKt*_NN *ko*_CM]_NP [*1990*_CD *mein*_CM]_NP [*takreeban*_RB]_RBP [*1200_CD karon*_NN *par*_CM]_NP [*mtwazn*_JJ]_JJP [*rakhne*_VBI *ki*_CM]_VGNN [*tawaqqo*_NN]_NP [*karte*_VB *hai*_AUXT]_VGF

Example 5

In example 5, *"said"* corresponds to one Urdu word *"کہا"(kaha)* that also captures the tense information (past). However, consider the verb *"expects"*. This is a clear case of noun-verb complex predicate where *"expects"* is mapped to *"توقع کرتی ہے"(tawaqqo karte hai)*.

ENG- *Not*_RB *all*_PDT *those*_DT *who*_WP ***wrote***_VBD ***oppose***_VBP *the*_DT *changes*_NNS
URD -*wo tamaam jinhon ne* ***likha*** *tabdeeliyon ke* [***mukhalif***_JJ]_JJP [*nahi*_RB]_RBP [***hain***_VBT]_VGF

Example 6

In example 6, verb predicates are *"wrote"* and *"oppose"*. Consider the word *"oppose"*. There are two ways of representing this word in Urdu. As a verb chunk the translation would be *"**mukhalifat** nahi karte"* and as an adjectival chunk *"**mukhalif** nahi hai"*. The latter form of representation is used widely in the available translation corpus. The Urdu equivalent *of "oppose"* is *"مخالف ہیں"(mukhalif hai)*.

Another interesting observation in example 6 is the existence of discontinuous predicates. Though *"oppose"* is one word in English, the Urdu representation has two words that do not occur together. The adverb *"nahi" ~"not"* occurs between the adjective and the verb. Statistically dealing with this issue is extremely challenging and affects the boundaries of other arguments. Generalizing the rules needed to identify discontinuous predicates requires more detailed analysis of the corpus – from the linguistic aspect – and has not been attempted in this paper. We however map *"مخالف نہیں ہیں"(mukhalif nahi hai)* to the predicate *"oppose"*. *"nahi"* is treated as an argument ARG_NEG in PropBank.

4 Projection Results

It is impossible for us to report our projection results on the entire data set as we do not have it manually annotated. For the purpose of evaluation, we manually annotated 100 long sentences (L) and 100 short sentences (S) from the full 2350 sentence set. All the results are reported on

this 200 set of sentences. Set L has sentences that each has more than two verb predicates and several arguments. The number of words per sentence here is greater than 55. S; on the other hand has sentences with about 40 words each and no complex SOV structures.

The results shown in Table 3 are for all tags (verbs+args) that are projected from English onto Urdu. In order to understand why the performance over L dips, consider the results in Table 4 that are for verb projections only. Some long sentences in English have Urdu translations that do not maintain the same structure. For example an English phrase – *"... might **prompt** individuals to **get** out of stocks altogether"* is written in Urdu in a way that the English representation would be *"what makes individuals to **get** out of stocks is ..."*. The Urdu equivalent word for *"prompt"* is missing and the associated lemma gets assigned to the Urdu equivalent of *"get"* (the next lemma). This also affects the argument projections. Another reason is the effect of word alignments itself. Clearly longer sentences have greater alignment errors.

All tags[8]	100 long sentences	100 short sentences
Actual Tags	1267	372
Correct Tags	943	325
Found Tags	1212	353
L : Precision 77.8% Recall 74.4% F-Score 76%		
S: Precision 92% Recall 87.4% F-Score 89.7%		

Table 3: when all tags are considered

Comparing the results of Table 4 to Table 3, we see that argument projections affect the recall. This is because the projections of arguments depend not only on the word alignments but also on the verb predicates. Incorrect verb predicates affect the argument projections.

Only lemma	100 long sentences	100 short sentences
Actual Tags	670	240
Correct Tags	490	208
Overall Tags	720	257
L: Precision 68% Recall 73.1% F-Score 70.45%		
S : Precision 80.9% Recall 86.6% F-Score 83.65%		

Table 4: for verb projections only

Table 5 summarizes the results obtained when only the word alignments are considered to project all tags. But when virtual phrase boundaries in English are also considered, the F-score improves by 8% (Table 6). This is because virtual boundaries in a way mark context switch and when considered in proximity to the word alignments yield better predicate boundaries.

100 long sentences : only alignments	
Actual Tags	1267
Correct Tags	617
Overall Tags	782
Precision 78.9% Recall 48.7% F-Score 60.2%	

Table 5: with only word alignments

100 long sentences : alignments + virtual boundaries	
Actual Tags	1267
Correct Tags	792
Overall Tags	1044
Precision 75.8% Recall 62.5% F-Score 68.5%	

Table 6: with word alignments and virtual boundaries

100 Sentences	ARG 0	ARG 1	ARG 2	ARG 3	ARG M
Long	124	271	67	25	140
Found	111	203	36	12	114
P %	89.5	74.9	53.7	48	81.42
Short	34	47	4	2	19
Found	30	45	4	2	19
P %	88.2	95.7	100	100	100

Table 7: results of argument projections
Precision (P) on arguments

Table 7 shows the results of argument projections over the first 4 arguments of PropBank – ARG0, ARG1, ARG2 and ARG3 (out of 24 arguments, majority are sparse in our test set) and the adjunct tag set ARGM.

5 Automatic Detection

The size of SRL annotated corpus generated for Urdu is limited with only 2350 sentences. To explore the possibilities of augmenting this data set, we train verb predicate and argument detection models. The results show great promise in generating large-scale automatic annotations.

5.1 Verb Predicate Detection

Verb predicate detection happens in two stages. In the first stage, the predicate boundaries are marked using a CRF (Lafferty *et al.*, 2001) based sequence labeling approach. The training data for the model is generated by annotating the automatically annotated Urdu SRL corpus using BI

[8] Tags - lemma (verb predicates) + arguments, Actual tags – number of tags in the English set, Found tags – number of tags transferred to Urdu, Correct Tags – number of tags correctly transferred

annotations. E.g. *kam* B-VG, k*arne par* I-VG. The non-verb predicates are labeled "-1". The model uses POS, chunk and lexical information as features. We report the results on a set of 77 sentences containing a mix of short and long sentences.

Number of verb predicates correctly marked	377
Num of verb predicates found	484
Actual num of verb predicates	451
Precision 77.8% Recall 83.5% F-Score 80.54%	

Table 8: CRF results for verb boundaries

Every verb predicate is associated with a lemma mapped from the English VerbNet map file[9]. E.g. the Urdu verb *"کم کرنے پر" (kam karne par)* has the lemma *"lower"*. The second stage includes assigning these lemmas. Lemma assignment is based on lookups from a VerbNet like map file. We have compiled a large set of Urdu verb predicates by mapping translations found in the automatically annotated corpus to the VerbNet map file. This Urdu verb predicate list also accommodates complex predicates that occur along with verbs such as *"karna – to do", "paana – to get",* etc. (along with different variations of these verbs – *karte, kiya, paate etc.*). This verb predicate list (manually corrected) consists of 800 entries. Since our gold standard test set is very small, the lemma assignment for all verb predicates is 100% (no **pb** values and hence no senses). This list, however, has to be augmented further to meet the standards of the English VerbNet map file.

5.2 Argument Detection

Argument detection (SRL) is done in two steps: (1) argument boundary detection (2) argument label assignment. We perform tests for step 2 to show how well a standard SVM role detection model works on the automatically generated Urdu data set. For each pair of correct predicate *p* and an argument *i* we create a feature representation $F_{p,a}$ ~ set T of all arguments. To train a multi-class role-classifier, given the set T of all arguments, T can be rationalized as $T^{+}_{\arg i}$ (positive instances) and $T^{-}_{\arg i}$ (negative instances) for each argument *i*. In this way, individual ONE-vs-ALL (Gildea and Jurafsky, 2002) classifier for each argument *i* is trained. In the testing phrase, given an unseen sentence, for each argument $F_{p,q}$ is generated and classified by each individual classifier.

We created a set of *standard* SRL features as shown in table 9. The results (Tables 10 and 11), though not impressive, are promising. We believe that by increasing the number of samples (for each argument) in the training set and intelligently controlling the negative samples, the results can be improved significantly.

Training – 2270 sentences with 7315 argument instances. Test – 77 sentences with 496 argument instances. (22 different role types)

BaseLine Features (BL)	phrase-type (syntactic category; NP, PP etc.), predicate (in our case, verb group), path (syntactic path from the argument constituent to the predicate), head words (argument and the predicate respectively), position (whether the phrase is before or after the predicate)
Detailed Features	BL + POS (of the first word in the predicate), chunk tag of the predicate, POS (of the first word of the constituent argument), head word (of the verb group in a complex predicate), named entity (whether the argument contains any named entity, such as location, person, organization etc.)

Table 9: Features for SRL

Kernel/features	Precision	Recall	F-Score
LK – BL	71.88	48.25	57.74
LK – all	73.91	47.55	57.87
PK – BL	74.19	48.25	58.47
PK –all (best)	73.47	49.65	59.26

Table 10: Arg0 performance

Kernel/features	Precision	Recall	F-Score
LK – BL	69.35	22.87	34.40
LK – all	69.84	23.4	35.05
PK – BL	73.77	24.14	36.38
PK –all (best)	73.8	26.06	38.52

Table 11: Arg1 Performances
(PK - polynomial kernel LK – Linear kernel)

6 Conclusion

In this work, we develop an alignment system that is tailor made to fit the SRL problem scope for Urdu. Furthermore, we have shown that despite English being a totally different language, resources for Urdu can be generated if the subtle grammatical nuances of Urdu are accounted for while projecting the annotations. We plan to work on argument boundary detection and explore other features for argument detection. The lemma set generated for Urdu is being refined for finer granularity.

[9] http://verbs.colorado.edu/semlink/semlink1.1/vn-pb/README.TXT

References

Ambati, Vamshi and Chen, Wei, 2007. Cross Lingual Syntax Projection for Resource-Poor Languages. CMU.

Baker, Collin .F., Charles J. Fillmore, John B. Lowe. 1998. The Berkeley Frame Net project. *COLING-ACL*.

Bharati, Akshar, Dipti Misra Sharma, Lakshmi Bai and Rajeev Sangal. 2006. AnnCorra: Annotating Corpora Guidelines For POS And Chunk Annotation For Indian Language. *Technical Report*, Language Technologies Research Centre IIIT, Hyderabad.

Burchardt, Aljoscha and Anette Frank. 2006. Approaching textual entailment with LFG and FrameNet frames. *RTE-2 Workshop*. Venice, Italy.

Butt, Miriam and Wilhelm Geuder. 2001. On the (semi)lexical status of light verbs. *Norbert Corver and Henk van Riemsdijk, (Eds.), Semi-lexical Categories: On the content of function words and the function of content words*, Mouton de Gruyter, pp. 323–370, Berlin.

Diab, Mona and Philip Resnik. 2002. An unsupervised method for word sense tagging using parallel corpora. *40th Annual Meeting of ACL*, pp. 255-262, Philadelphia, PA.

Dorr, Bonnie, J. 1994. Machine Translation Divergences: A Formal Description and Proposed Solution. *ACL,* Vol. 20(4), pp. 597-631.

Fillmore, Charles J. 1968. The case for case. *Bach, & Harms(Eds.), Universals in Linguistic Theory*, pp. 1-88. Holt, Rinehart, and Winston, New York.

Fillmore, Charles J. 1982. Frame semantics. *Linguistics in the Morning Calm*, pp.111-137. Hanshin, Seoul, S. Korea.

Fung, Pascale and Benfeng Chen. 2004. BiFrameNet: Bilingual frame semantics resources construction by cross-lingual induction. *20th International Conference on Computational Linguistics*, pp. 931-935, Geneva, Switzerland.

Fung, Pascale, Zhaojun Wu, Yongsheng Yang and Dekai Wu. 2007. Learning bilingual semantic frames: Shallow semantic parsing vs. semantic role projection. *11th Conference on Theoretical and Methodological Issues in Machine Translation*, pp. 75-84, Skovde, Sweden.

Gildea, Daniel and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, Vol. 28(3), pp. 245-288.

Hi, Chenhai and Rebecca Hwa. 2005. A backoff model for bootstrapping resources for non-english languages. *Joint Human Language Technology Conference and Conference on EMNLP*, pp. 851-858, Vancouver, BC.

Hwa, Rebecca, Philip Resnik, Amy Weinberg, and Okan Kolak. 2002. Evaluation translational correspondance using annotation projection. *40th Annual Meeting of ACL*, pp. 392-399, Philadelphia, PA.

Koehn, Phillip. 2005. "Europarl: A parallel corpus for statistical machine translation," MT summit, Citeseer.

Lafferty, John D., Andrew McCallum and C.N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. *18th International Conference on Machine Learning*, pp. 282-289.

Liang, Percy, Ben Taskar, and Dan Klein. 2006. Alignment by Agreement, *NAACL*.

Marcus, Mitchell P., Beatrice Santorini and Mary Ann Marcinkiewicz. 2004. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, Vol. 19(2), pp. 313-330.

Moschitti, Alessandro. 2008. Kernel methods, syntax and semantics for relational text categorization. *17th ACM CIKM*, pp. 253-262, Napa Valley, CA.

Mukerjee, Amitabh , Ankit Soni and Achala M. Raina. 2006. Detecting Complex Predicates in Hindi using POS Projection across Parallel Corpora. *Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*, pp. 11–18. Sydney.

Mukund, S., Srihari, R. K., and Peterson, E. 2010. An Information Extraction System for Urdu – A Resource Poor Language. *Special Issue on Information Retrieval for Indian Languages*. TALIP.

Pado, Sebastian and Mirella Lapata. 2009. Cross-Lingual annotation Projection of Semantic Roles. *Journal of Artificial Intelligence Research*, Vol. 36, pp. 307-340.

Palmer, Martha, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, Vol. 31(1).

Palmer, Martha, Shijong Ryu, Jinyoung Choi, Sinwon Yoon, and Yeongmi Jeon. 2006. Korean Propbank. *Linguistic data consortium*, Philadelphia.

Philips, Lawrence. 2000. The Double Metaphone Search Algorithm. *C/C++ Users Journal*.

Sinha, R. Mahesh K. 2009. Mining Complex Predicates In Hindi Using A Parallel Hindi-English Corpus. *ACL International Joint Conference in Natural Language Processing*, pp 40.

Surdeanu, Mihai, Sanda Harabagiu, John Williams and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. *41st Annual Meeting of the Association for Computational Linguistics*, pp. 8-15, Sapporo, Japan.

Taule, Mariona, M. Antonio Marti, and Marta Recasens. 2008. Ancora: Multi level annotated corpora for Catalan and Spanish. *6th International Conference on Language Resources and Evaluation*, Marrakesh, Morocco.

Xue, Nianwen and Martha Palmer. 2009. Adding semantic roles to the Chinese treebank. *Natural Language Engineering*, Vol. 15(1), pp. 143-172.

Yarowsky, David, Grace Ngai and Richard Wicentowski. 2001. Inducing multi lingual text analysis tools via robust projection across aligned corpora. *1st Human Language Technology Conference*, pp. 161-168, San Francisco, CA.

Recognition of Affect, Judgment, and Appreciation in Text

Alena Neviarouskaya
University of Tokyo
lena@mi.ci.i.u-tokyo.ac.jp

Helmut Prendinger
Nat. Institute of Informatics Tokyo
helmut@nii.ac.jp

Mitsuru Ishizuka
University of Tokyo
ishizuka@i.u-tokyo.ac.jp

Abstract

The main task we address in our research is classification of text using fine-grained attitude labels. The developed @AM system relies on the compositionality principle and a novel approach based on the rules elaborated for semantically distinct verb classes. The evaluation of our method on 1000 sentences, that describe personal experiences, showed promising results: average accuracy on the fine-grained level (14 labels) was 62%, on the middle level (7 labels) – 71%, and on the top level (3 labels) – 88%.

1 Introduction and Related Work

With rapidly growing online sources aimed at encouraging and stimulating people's discussions concerning personal, public or social issues (news, blogs, discussion forums, etc.), there is a great need in development of a computational tool for the analysis of people's attitudes. According to the Appraisal Theory (Martin and White, 2005), attitude types define the specifics of appraisal being expressed: affect (personal emotional state), judgment (social or ethical appraisal of other's behaviour), and appreciation (evaluation of phenomena).

To analyse contextual sentiment of a phrase or a sentence, rule-based approaches (Nasukawa and Yi, 2003; Moilanen and Pulman, 2007; Subrahmanian and Reforgiato, 2008), a machine-learning method using not only lexical but also syntactic features (Wilson et al., 2005), and a model of integration of machine learning approach with compositional semantics (Choi and Cardie, 2008) were proposed. With the aim to recognize fine-grained emotions from text on the level of distinct sentences, researchers have employed a keyword spotting technique (Chuang and Wu, 2004; Strapparava et al., 2007), a technique calculating emotion scores using Pointwise Mutual Information (PMI) (Kozareva et al., 2007), an approach inspired by common-sense knowledge (Liu et al., 2003), rule-based linguistic approaches (Boucouvalas, 2003; Chaumartin, 2007), machine-learning methods (Alm, 2008; Aman and Szpakowicz, 2008; Strapparava and Mihalcea, 2008), and an ensemble based multi-label classification technique (Bhowmick et al., 2009).

Early attempts to focus on distinct attitude types in the task of attitude analysis were made by Taboada and Grieve (2004), who determined a potential value of adjectives for affect, judgement and appreciation by calculating the PMI with the pronoun-copular pairs '*I was (affect)*', '*He was (judgement)*', and '*It was (appreciation)*', and Whitelaw et al. (2005), who used a machine learning technique (SVM) with fine-grained semantic distinctions in features (attitude type, orientation) in combination with "bag of words" to classify movie reviews. However, the concentration only on adjectives expressing appraisal and their modifiers greatly narrows the potential of the Whitelaw et al. (2005) approach.

In this paper we introduce our system @AM (**AT**titude **A**nalysis **M**odel), which (1) classifies sentences according to the fine-grained attitude labels (nine affect categories (Izard, 1971): 'anger', 'disgust', 'fear', 'guilt', 'interest', 'joy', 'sadness', 'shame', 'surprise'; four polarity labels for judgment and appreciation: 'POS jud', 'NEG jud', 'POS app', 'NEG app'; and 'neutral'); (2) assigns the strength of the attitude; and (3) determines the level of confidence, with which the attitude is expressed. @AM relies on a compositionality principle and a novel approach

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 806–814,
Beijing, August 2010

based on the rules elaborated for semantically distinct verb classes.

2 Lexicon for Attitide Analysis

We built a lexicon for attitude analysis that includes: (1) attitude-conveying terms; (2) modifiers; (3) "functional" words; and (4) modal operators.

2.1 The Core of Lexicon

As a core of lexicon for attitude analysis, we employ an Affect database and extended version of the SentiFul database developed by Neviarouskaya et al. (2009). The affective features of each emotion-related word are encoded using nine emotion labels ('anger', 'disgust', 'fear', 'guilt', 'interest', 'joy', 'sadness', 'shame', and 'surprise') and corresponding emotion intensities that range from 0.0 to 1.0. The original version of SentiFul database, which contains sentiment-conveying adjectives, adverbs, nouns, and verbs annotated by sentiment polarity, polarity scores and weights, was manually extended using attitude labels. Some examples of annotated attitude-conveying words are listed in Table 1. It is important to note here that some words may express different attitude types (affect, judgment, appreciation) depending on context; such lexical entries were annotated by all possible categories.

POS	Word	Category	Intensity
adjective	*honorable*	POS jud	0.3
	unfriendly	NEG aff (sadness)	0.5
		NEG jud	0.5
		NEG app	0.5
adverb	*gleefully*	POS aff (joy)	0.9
noun	*abnormality*	NEG app	0.25
verb	*frighten*	NEG aff (fear)	0.8
	desire	POS aff (interest)	1.0
		POS aff (joy)	0.5

Table 1. Examples of attitude-conveying words and their annotations.

2.2 Modifiers and Functional Words

We collected 138 modifiers that have an impact on contextual attitude features of related words, phrases, or clauses. They include:

1. Adverbs of degree (e.g., '*significantly*', '*slightly*' etc.) and affirmation (e.g., '*absolutely*', '*seemingly*') that have an influence on the strength of the attitude of related words. Two annotators gave coefficients for intensity degree strengthening or weakening (from 0.0 to 2.0) to each adverb, and the result was averaged (e.g., coeff('*slightly*') = 0.2).
2. Negation words (e.g., '*never*', '*nothing*' etc.) reversing the polarity of related statement.
3. Adverbs of doubt (e.g., '*scarcely*', '*hardly*' etc.) and falseness (e.g., '*wrongly*' etc.) reversing the polarity of related statement.
4. Prepositions (e.g., '*without*', '*despite*' etc.) neutralizing the attitude of related words.
5. Condition operators (e.g., '*if*', '*even though*' etc.) that neutralize the attitude of related words.

We distinguish two types of "functional" words that influence contextual attitude and its strength:

1. *Intensifying* adjectives (e.g., '*rising*', '*rapidly-growing*'), nouns (e.g., '*increase*'), and verbs (e.g., '*to grow*', '*to rocket*') that increase the strength of attitude of related words.
2. *Reversing* adjectives (e.g., '*reduced*'), nouns (e.g., '*termination*), and verbs (e.g., '*to decrease*', '*to limit*', '*to diminish*'), which reverse the prior polarity of related words.

2.3 Modal Operators

Consideration of the modal operators in the tasks of opinion mining and attitude analysis is very important, as they indicate a degree of person's belief in the truth of the proposition, which is subjective in nature (Hoye, 1997). Modals are distinguished by their *confidence level*. We collected modal operators of two categories: modal verbs (13 verbs) and modal adverbs (61 adverbs). Three human annotators assigned the *confidence level* ranging from 0.0 to 1.0 to each modal verb and adverb; these ratings were averaged (e.g., conf('*vaguely*') = 0.17, conf('*arguably*') = 0.63, conf('*would*') = 0.8, conf('*veritably*') = 1.0).

3 Compositionality Principle

Our algorithm for attitude classification is designed based on the *compositionality principle*, according to which we determine the attitudinal meaning of a sentence by composing the pieces that correspond to lexical units or other linguistic constituent types governed by the rules of *polarity reversal, aggregation (fusion)*, *propagation*, *domination*, *neutralization*, and *intensification*, at various grammatical levels.

Polarity reversal means that a phrase or statement containing an attitude-conveying

term/phrase with prior positive polarity becomes negative, and vice versa. The rule of *polarity reversal* is applied in three cases: (1) negation word-modifier in relation with an attitude-conveying statement (e.g., '*never*' & POS('*succeed*') => NEG('*never succeed*')); (2) adverb of doubt in relation with attitude-conveying statement (e.g., '*scarcely*' & POS('*relax*') => NEG('*scarcely relax*')); (3) functional word of *reversing* type in relation with attitude-conveying statement (e.g., adjective '*reduced*' & POS('*enthusiasm*') => NEG('*reduced enthusiasm*')). In the case of judgment and appreciation, the use of the *polarity reversal* rule is straightforward ('POS jud' <=> 'NEG jud', 'POS app' <=> 'NEG app'). However, it is not trivial to find pairs of opposite emotions in the case of a fine-grained classification, except for 'joy' and 'sadness'. Therefore, we assume that (1) the opposite emotion for three positive emotions, i.e. 'interest', 'joy', and 'surprise', is 'sadness' ('POS aff' => 'sadness'); and (2) the opposite emotion for six negative emotions, i.e. 'anger', 'disgust', 'fear', 'guilt', 'sadness', and 'shame', is 'joy' ('NEG aff' => 'joy').

The rules of *aggregation (fusion)* are as follows: (1) if polarities of attitude-conveying terms in adjective-noun, noun-noun, adverb-adjective, adverb-verb phrases have opposite directions, mixed polarity with dominant polarity of a pre-modifier is assigned to the phrase (e.g., POS('*beautiful*') & NEG('*fight*') => POS-neg('*beautiful fight*'); NEG('*shamelessly*') & POS('*celebrate*') => NEG-pos('*shamelessly celebrate*')); otherwise (2) the resulting polarity is based on the equal polarities of terms, and the strength of attitude is measured as a maximum between polarity scores (intensities) of terms (*max*(score1,score2)).

The rule of *propagation* is useful, as proposed in (Nasukawa and Yi, 2003), for the task of the detection of local sentiments for given subjects. "Propagation" verbs propagate the sentiment towards the arguments; "transfer" verbs transmit sentiments among the arguments. The rule of *propagation* is applied when a verb of "propagation" or "transfer" type is used in a phrase/clause and sentiment of an argument that has prior neutral polarity needs to be investigated (e.g., PROP-POS('*to admire*') & '*his behaviour*' => POS('*his behaviour*'); '*Mr. X*' & TRANS('*supports*') & NEG('*crime business*') => NEG('*Mr. X*')).

The rules of *domination* are as follows: (1) if polarities of a verb (this rule is applied only for certain classes of verbs) and an object in a clause have opposite directions, the polarity of verb is prevailing (e.g., NEG('*to deceive*') & POS('*hopes*') => NEG('*to deceive hopes*')); (2) if compound sentence joints clauses using coordinate connector '*but*', the attitude features of a clause following after the connector are dominant (e.g., 'NEG(*It was hard to climb a mountain all night long*)*, but* POS(*a magnificent view rewarded the traveler at the morning*).' => POS(whole sentence)).

The rule of *neutralization* is applied when preposition-modifier or condition operator relate to the attitude-conveying statement (e.g., '*despite*' & NEG('*worries*') => NEUT('*despite worries*')).

The rule of *intensification* means strengthening or weakening of the polarity score (intensity), and is applied when:

1. adverb of degree or affirmation relates to attitude-conveying term (e.g., Pos_score('*happy*') < Pos_score('*extremely happy*'));
2. adjective or adverb is used in a comparative or superlative form (e.g., Neg_score('*sad*') < Neg_score('*sadder*') < Neg_score ('*saddest*')).

Our method is capable of processing sentences of different complexity, including simple, compound, complex (with complement and relative clauses), and complex-compound sentences. We employ Connexor Machinese Syntax parser (`http://www.connexor.eu/`) that returns lemmas, parts of speech, dependency functions, syntactic function tags, and morphological tags. When handling the parser output, we represent the sentence as a set of primitive clauses. Each clause might include Subject formation, Verb formation and Object formation, each of which may consist of a main element (subject, verb, or object) and its attributives and complements. For the processing of complex or compound sentences, we build a so-called "relation matrix", which contains information about dependences (e.g., coordination, subordination, condition, contingency, etc.) between different clauses in a sentence. While applying the compositionality principle, we consecutively assign attitude fea-

tures to words, phrases, formations, clauses, and finally, to the whole sentence.

4 Consideration of the Semantics of Verbs

All sentences must include a verb, because the verb tells us what action the subject is performing and object is receiving. In order to elaborate rules for attitude analysis based on the semantics of verbs, we investigated VerbNet (Kipper et al., 2007), the largest on-line verb lexicon that is organized into verb classes characterized by syntactic and semantic coherence among members of a class. Based on the thorough analysis of 270 first-level classes of VerbNet and their members, 73 verb classes (1) were found useful for the task of attitude analysis, and (2) were further classified into 22 classes differentiated by the role that members play in attitude analysis and by rules applied to them. Our classification is shown in Table 2.

For each of our verb classes, we developed set of rules that are applied to attitude analysis on the phrase/clause-level. Some verb classes (e.g., "*Psychological state or emotional reaction*", "*Judgment*", "*Bodily state and damage to the body*", "*Preservation*" etc.) include verbs annotated by attitude type, prior polarity orientation, and the strength of attitude. The attitude features of phrases that involve positively or negatively charged verbs from such classes are context-sensitive and are defined by means of rules designed for each of the class.

As an example, we provide short description and rules elaborated for the subclass "*Object-centered (oriented) emotional state*".

Features: subject experiences emotions towards some stimulus; verb prior polarity: positive or negative; context-sensitive.

Verb-Object rules (subject is ignored):

1. "Interior perspective" (subject's inner emotion state or attitude):

S & V+('*admires*') & O+('*his brave heart*') => (fusion, *max*(V_score,O_score)) => 'POS aff'.

S & V+('*admires*') & O-('*mafia leader*') => (verb valence dominance, V_score) => 'POS aff'.

S & V-('*disdains*') & O+('*his honesty*') => (verb valence dominance, V_score) => 'NEG aff'.

Verb class (verb samples)
1 Psychological state or emotional reaction
1.1 Object-centered (oriented) emotional state (*adore*)
1.2 Subject-driven change in emotional state (trans.) (*charm*, *inspire*, *bother*)
1.3 Subject-driven change in emotional state (intrans.) (*appeal to*, *grate on*)
2 Judgment
2.1 Positive judgment (*bless*, *honor*)
2.2 Negative judgment (*blame*, *punish*)
3 Favorable attitude (*accept*, *allow*, *tolerate*)
4 Adverse (unfavorable) attitude (*discourage*, *forbid*)
5 Favorable or adverse calibratable changes of state (*grow*, *decline*)
6 Verbs of removing
6.1 Verbs of removing with neutral charge (*delete*)
6.2 Verbs of removing with negative charge (*expel*)
6.3 Verbs of removing with positive charge (*evacuate*)
7 Negatively charged change of state (*break*, *crush*)
8 Bodily state and damage to the body (*sicken*, *injure*)
9 Aspectual verbs
9.1 Initiation, continuation of activity, and sustaining (*begin*, *continue*, *maintain*)
9.2 Termination of activity (*quit*, *finish*)
10 Preservation (*defend*, *insure*)
11 Verbs of destruction and killing (*damage*, *poison*)
12 Disappearance (*disappear*, *die*)
13 Limitation and subjugation (*confine*, *restrict*)
14 Assistance (*succor*, *help*)
15 Obtaining (*win*, *earn*)
16 Communication indicator/reinforcement of attitude (*guess*, *complain*, *deny*)
17 Verbs of leaving (*abandon*, *desert*)
18 Changes in social status or condition (*canonize*)
19 Success and failure
19.1 Success (*succeed*, *manage*)
19.2 Failure (*fail*, *flub*)
20 Emotional nonverbal expression (*smile*, *weep*)
21 Social interaction (*marry*, *divorce*)
22 Transmitting verbs (*supply*, *provide*)

Table 2. Verb classes for attitude analysis.

S & V-('*disdains*') & O-('*criminal activities*') => (fusion, *max*(V_score,O_score)) => 'NEG aff'.

2. "Exterior perspective" (social/ethical judgment):

S & V+('*admires*') & O+('*his brave heart*') => (fusion, *max*(V_score,O_score)) => 'POS jud'.

S & V+('*admires*') & O-('*mafia leader*') => (verb valence reversal, *max*(V_score,O_score)) => 'NEG jud'.

S & V-('*disdains*') & O+('*his honesty*') => (verb valence dominance, *max*(V_score,O_score)) => 'NEG jud'.

S & V-('*disdains*') & O-('*criminal activities*') => (verb valence reversal, *max*(V_score,O_score)) => 'POS jud'.

3. In case of neutral object => attitude type and prior polarity of verb, verb score (V_score).

Verb-PP (prepositional phrase) rules:

1. In case of negatively charged verb and PP starting with '*from*' => verb dominance:

S & V-('*suffers*') & PP-('*from illness*') => interior: 'NEG aff'; exterior: 'NEG jud'.

S & V-('*suffers*') & PP+ ('*from love*') => interior: 'NEG aff'; exterior: 'NEG jud'.

2. In case of positively charged verb and PP starting with '*in*'/'*for*' => treat PP the same way as object (see above):

S & V+('*believes*') & PP-('*in evil*') => interior: 'POS aff'; exterior: 'NEG jud'.

S & V+('*believes*') & PP+('*in kindness*') => interior: 'POS aff'; exterior: 'POS jud'.

In the majority of rules the strength of attitude is measured as a maximum between attitude scores (for example, the attitude conveyed by '*to suffer from grave illness*' is stronger than that of '*to suffer from slight illness*').

In contrast to the rules of "*Object-centered (oriented) emotional state*" subclass, which ignore attitude features of a subject in a sentence, the rules elaborated for the "*Subject-driven change in emotional state (trans.)*" disregard the attitude features of object, as in sentences involving members of this subclass object experiences emotion, and subject causes the emotional state. For example (due to limitation of space, here and below we provide only some cases):

S('*Classical music*') & V+('*calmed*') & O-('*disobedient child*') => interior: 'POS aff'; exterior: 'POS app'.

S-('*Fatal consequences of GM food intake*') & V-('*frighten*') & O('*me*') => interior: 'NEG aff'; exterior: 'NEG app'.

The Verb-Object rules for the "*Judgment*" subclasses, namely "*Positive judgment*" and "*Negative judgment*", are very close to those defined for the subclass "*Object-centered (oriented) emotional state*". However, Verb-PP rules have some specifics: for both positive and negative judgment verbs, we treat PP starting with '*for*'/'*of*'/'*as*' the same way as object in Verb-Object rules. For example:

S('*He*') & V-('*blamed*') & O+('*innocent person*') => interior: 'NEG jud'; exterior: 'NEG jud'.

S('*They*') & V-('*punished*') & O('*him*') & PP-('*for his misdeed*') => interior: 'NEG jud'; exterior: 'POS jud'.

Verbs from classes "*Favorable attitude*" and "*Adverse (unfavorable) attitude*" have prior neutral polarity and positive or negative reinforcement, correspondingly, that means that they only impact on the polarity and strength of non-neutral phrase (object in a sentence written in active voice, or subject in a sentence written in passive voice, or PP in case of some verbs). The rules are:

1. If verb belongs to the "*Favorable attitude*" class and the polarity of phrase is not neutral, then the attitude score of the phrase is intensified (symbol '^' means intensification):

S('*They*') & [V pos. reinforcement]('*elected*') & O+('*fair judge*') => 'POS app'; O_score^.

S('*They*') & [V pos. reinforcement]('*elected*') & O-('*corrupt candidate*') => 'NEG app'; O_score^.

2. If verb belongs to the "*Adverse (unfavorable) attitude*" class and the polarity of phrase is not neutral, then the polarity of phrase is reversed and score is intensified:

S('*They*') & [V neg. reinforcement]('*prevented*') & O-('*the spread of disease*') => 'POS app'; O_score^.

S+('*His achievements*') & [V neg. reinforcement]('*were overstated*') => 'NEG app'; S_score^.

Below are examples of processing the sentences with verbs from "*Verbs of removing*" class.

"*Verbs of removing with neutral charge*":

S('*The tape-recorder*') & [V neutral rem.]('*automatically ejects*') & O-neutral('*the tape*') => neutral.

S('*The safety invention*') & [V neutral rem.]('*ejected*') & O('*the pilot*') & PP-('*from burning plane*') => 'POS app'; PP_score^.

"*Verbs of removing with negative charge*":

S('*Manager*') & [V neg. rem.]('*fired*') & O-('*careless employee*') & PP('*from the company*') => 'POS app'; *max*(V_score,O_score).

"*Verbs of removing with positive charge*":

S('*They*') & [V pos. rem.]('*evacuated*') & O('*children*') & PP-('*from dangerous place*') => 'POS app'; *max*(V_score,PP_score).

Along with modal verbs and modal adverbs, members of the "*Communication indicator/reinforcement of attitude*" verb class also in-

dicate the confidence level or degree of certainty concerning given opinion. Features are: subject (communicator) expresses statement with/without attitude; statement is PP starting with '*of*', '*on*', '*against*', '*about*', '*concerning*', '*regarding*', '*that*', '*how*' etc.; ground: positive or negative; reinforcement: positive or negative. The rules are:

1. If the polarity of expressed statement is neutral, then the attitude is neutral:

S('*Professor*') & [V pos. ground, pos. reinforcement, confidence:0.83]('*dwelled*') & PP-neutral('*on a question*') => neutral.

2. If the polarity of expressed statement is not neutral and the reinforcement is positive, then the score of the statement (PP) is intensified:

S('*Jane*') & [V neg. ground, pos. reinforcement, confidence:0.8]('*is complaining*') & PP-('*of a headache again*') => 'NEG app'; PP_score^; confidence:0.8.

3. If the polarity of expressed statement is not neutral and reinforcement is negative, then the polarity of the statement (PP) is reversed and score is intensified:

S('*Max*') & [V neg. ground, neg. reinforcement, confidence:0.2]('*doubt*') & PP-{'*that*' S+('*his good fortune*') & [V termination]('*will ever end*')} => 'POS app'; PP_score^; confidence:0.2.

In the last example, to measure the sentiment of PP, we apply rule for the verb '*end*' from the "*Termination of activity*" class, which reverses the non-neutral polarity of subject (in intransitive use of verb) or object (in transitive use of verb). For example, the polarity of both sentences '*My whole enthusiasm and excitement disappear like a bubble touching a hot needle*' and '*They discontinued helping children*' is negative.

5 Decision on Attitude Label

The decision on the most appropriate final label for the clause, in case @AM annotates it using different attitude types according to the words with multiple annotations (e.g., see word '*unfriendly*' in Table 1) or based on the availability of the words conveying different attitude types, is made based on the analysis of:

1) morphological tags of nominal heads and their premodifiers in the clause (e.g., first person pronoun, third person pronoun, demonstrative pronoun, nominative or genitive noun, etc.);

2) the sequence of hypernymic semantic relations of a particular noun in WordNet (Miller, 1990), which allows to determine its conceptual domain (e.g., "person, human being", "artifact", "event", etc.);

3) the annotations from the Stanford Named Entity Recognizer (Finkel et al. 2005) that labels PERSON, ORGANIZATION, and LOCATION entities.

For ex., '*I feel highly unfriendly attitude towards me*' conveys emotion ('NEG aff': 'sadness'), while '*The shop assistant's behavior was really unfriendly*' and '*Plastic bags are environment unfriendly*' express judgment ('NEG jud') and appreciation ('NEG app'), correspondingly.

6 Evaluation

For the experiments, we used our own data set, as, to the best of our knowledge, there is no publicly available data set of sentences annotated by the fine-grained labels proposed in our work. In order to evaluate the performance of our algorithm, we created the data set of sentences extracted from personal stories about life experiences that were anonymously published on the *Experience Project* website (`www.experienceproject.com`), where people share personal experiences, thoughts, opinions, feelings, passions, and confessions through the network of personal stories. With over 4 million experiences accumulated (as of February 2010), *Experience Project* is a perfect source for researchers interested in studying different types of attitude expressed through text.

6.1 Data Set Description

For our experiment we extracted 1000 sentences[1] from various stories grouped by topics within 13 different categories, such as "Arts and entertainment", "Current events", "Education", "Family and friends", "Health and wellness", "Relationships and romance" and others, on the *Experience Project* website. Sentences were collected from 358 distinct topic groups, such as "I still remember September 11", "I am intelligent but airheaded", "I think bullfighting is cruel", "I quit smoking", "I am a fashion victim", "I was adopted" and others.

[1] This annotated data set is freely available upon request.

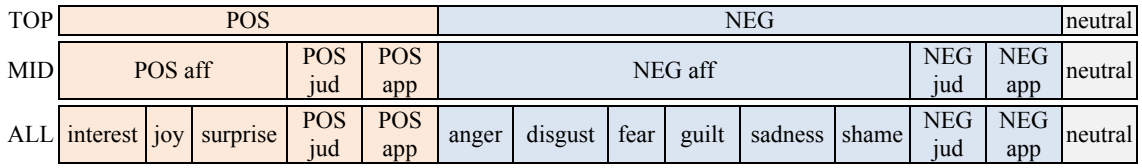


Figure 1. Hierarchy of attitude labels.

We considered three hierarchical levels of attitude labels in our experiment (see Figure 1). Three independent annotators labeled the sentences with one of 14 categories from the ALL level and a corresponding score (the strength or intensity value). These annotations were further interpreted using labels from the MID and the TOP levels. Fleiss' Kappa coefficient was used as a measure of reliability of human raters' annotations. The agreement coefficient on 1000 sentences was 0.53 on ALL level, 0.57 on MID level, and 0.73 on TOP level.

Only those sentences, on which at least two out of three human raters completely agreed, were included in the gold standards for our experiment. Three gold standards were created according to the hierarchy of attitude labels. Fleiss' Kappa coefficients are 0.62, 0.63, and 0.74 on ALL, MID, and TOP levels, correspondingly. Table 3 shows the distributions of labels in the gold standards.

ALL level	
Label	**Number**
anger	45
disgust	21
fear	54
guilt	22
interest	84
joy	95
sadness	133
shame	18
surprise	36
POS jud	66
NEG jud	78
POS app	100
NEG app	29
neutral	87
total	868

MID level	
Label	**Number**
POS aff	233
NEG aff	332
POS jud	66
NEG jud	78
POS app	100
NEG app	29
neutral	87
total	925

TOP level	
Label	**Number**
POS	437
NEG	473
neutral	87
total	997

Table 3. Label distributions in gold standards.

6.2 Results

The results of a simple method selecting the attitude label with the maximum intensity from the annotations of sentence tokens found in the database were considered as the baseline. After processing each sentence from the data set by the baseline method and our @AM system, we measured averaged accuracy, precision, recall, and F-score for each label in ALL, MID, and TOP levels. The results are shown in Table 4.

As seen from the obtained results, our algorithm performed with high accuracy significantly surpassing the baselines in all levels of attitude hierarchy, thus demonstrating the contribution of the sentence parsing and our hand-crafted rules to the reliable recognition of attitude from text. Two-tailed t-tests with significance level of 0.05 showed that the differences in accuracy between the baseline method and our @AM system are statistically significant ($p<0.001$) in fine-grained as well as coarse-grained classifications.

In the case of fine-grained attitude recognition (ALL level), the highest precision was obtained for 'shame' (0.923) and 'NEG jud' (0.889), while the highest recall was received for 'sadness' (0.917) and 'joy' (0.905) emotions at the cost of low precision (0.528 and 0.439, correspondingly). The algorithm performed with the worst results in recognition of 'NEG app' and 'neutral'.

The analysis of a confusion matrix for the ALL level revealed the following top confusions of our system: (1) 'anger', 'fear', 'guilt', 'shame', 'NEG jud', 'NEG app' and 'neutral' were predominantly incorrectly predicted as 'sadness' (for ex., @AM resulted in 'sadness' for the sentence '*I know we have several months left before the election, but I am already sick and tired of seeing the ads on TV*', while human annotations were 'anger'/'anger'/'disgust'); (2) 'interest', 'POS jud' and 'POS app' were mostly confused with 'joy' by our algorithm (e.g., @AM classified the sentence '*It's one of those life changing artifacts that we must have in order to have happier, healthier lives*' as 'joy'(-ful), while human annotations were 'POS app'/'POS app'/'interest').

Our system achieved high precision for all categories on the MID level (Table 4), with the exception of 'NEG app' and 'neutral', although

Level	Label	Baseline method				@AM			
		Accuracy	Precision	Recall	F-score	Accuracy	Precision	Recall	F-score
ALL	anger	0.437	0.742	0.511	0.605	**0.621**	0.818	0.600	0.692
	disgust		0.600	0.857	0.706		0.818	0.857	0.837
	fear		0.727	0.741	0.734		0.768	0.796	0.782
	guilt		0.667	0.364	0.471		0.833	0.455	0.588
	interest		0.380	0.357	0.368		0.772	0.524	0.624
	joy		0.266	0.579	0.364		0.439	0.905	0.591
	sadness		0.454	0.632	0.528		0.528	0.917	0.670
	shame		0.818	0.500	0.621		0.923	0.667	0.774
	surprise		0.625	0.694	0.658		0.750	0.833	0.789
	POS jud		0.429	0.227	0.297		0.824	0.424	0.560
	NEG jud		0.524	0.141	0.222		0.889	0.410	0.561
	POS app		0.349	0.150	0.210		0.755	0.400	0.523
	NEG app		0.250	0.138	0.178		0.529	0.310	0.391
	neutral		0.408	0.483	0.442		0.559	0.437	0.490
MID	POS aff	0.524	0.464	0.695	0.557	**0.709**	0.668	0.888	0.762
	NEG aff		0.692	0.711	0.701		0.765	0.910	0.831
	POS jud		0.405	0.227	0.291		0.800	0.424	0.554
	NEG jud		0.458	0.141	0.216		0.842	0.410	0.552
	POS app		0.333	0.150	0.207		0.741	0.400	0.519
	NEG app		0.222	0.138	0.170		0.474	0.310	0.375
	neutral		0.378	0.483	0.424		0.514	0.437	0.472
TOP	POS	0.732	0.745	0.796	0.770	**0.879**	0.918	0.920	0.919
	NEG		0.831	0.719	0.771		0.912	0.922	0.917
	neutral		0.347	0.483	0.404		0.469	0.437	0.452

Table 4. Results of the evaluation of performance of the baseline method and @AM system.

high recall was obtained only in the case of categories related to affect ('POS aff', 'NEG aff'). These results indicate that affect sensing is easier than recognition of judgment or appreciation from text. TOP level results (Table 4) show that our algorithm classifies sentences that convey positive or negative sentiment with high accuracy (92% and 91%, correspondingly). On the other hand, 'neutral' sentences still pose a challenge.

The analysis of errors revealed that system requires common sense or additional context to deal with sentences like '*All through my life I've felt like I'm second fiddle*' (gold standard: 'sadness'; @AM: 'neutral') or '*For me every minute on my horse is alike an hour in heaven!*' (gold standard: 'joy'; @AM: 'neutral').

We also evaluated the system performance with regard to attitude intensity estimation. The percentage of attitude-conveying sentences (not considering neutral ones), on which the result of our system conformed to the fine-grained gold standard (ALL level), according to the measured distance between intensities given by human raters (averaged values) and those obtained by our system is shown in Table 5. As seen from the table, our system achieved satisfactory results in estimation of the strength of attitude expressed through text.

Range of intensity difference	Percent of sentences, %
[0.0 – 0.2]	55.5
(0.2 – 0.4]	29.5
(0.4 – 0.6]	12.2
(0.6 – 0.8]	2.6
(0.8 – 1.0]	0.2

Table 5. Results on intensity.

7 Conclusions

In this paper we introduced @AM, which is so far, to the best of our knowledge, the only system classifying sentences using fine-grained attitude types, and extensively dealing with the semantics of verbs in attitude analysis. Our composition approach broadens the coverage of sentences with complex contextual attitude. The evaluation results indicate that @AM achieved reliable results in the task of textual attitude analysis. The limitations include dependency on lexicon and on accuracy of the parser. The primary objective for the future research is to develop a method for the extraction of reasons behind the expressed attitude.

References

Alm, Cecilia O. 2008. *Affect in Text and Speech*. PhD Dissertation. University of Illinois at Urbana-Champaign.

Aman, Saima, and Stan Szpakowicz. 2008. Using Roget's Thesaurus for Fine-Grained Emotion Recognition. *Proceedings of the Third International Joint Conference on Natural Language Processing*, Hyderabad, India, pp. 296-302.

Bhowmick, Plaban K., Anupam Basu, and Pabitra Mitra. 2009. Reader Perspective Emotion Analysis in Text through Ensemble based Multi-Label Classification Framework. *Computer and Information Science*, 2 (4): 64-74.

Boucouvalas, Anthony C. 2003. Real Time Text-to-Emotion Engine for Expressive Internet Communications. *Being There: Concepts, Effects and Measurement of User Presence in Synthetic Environments*, Ios Press, pp. 306-318.

Chaumartin, Francois-Regis. 2007. UPAR7: A Knowledge-based System for Headline Sentiment Tagging. *Proceedings of the SemEval-2007 International Workshop*, pp. 422-425.

Choi, Yejin, and Claire Cardie. 2008. Learning with Compositional Semantics as Structural Inference for Subsentential Sentiment Analysis. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 793-801.

Chuang, Ze-Jing, and Chung-Hsien Wu. 2004. Multimodal Emotion Recognition from Speech and Text. *Computational Linguistic and Chinese Language Processing*, 9(2): 45-62.

Finkel, Jenny R., Trond Grenager, and Christopher Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. *Proceedings of the 43nd Annual Meeting of the ACL*, pp. 363-370.

Hoye, Leo. 1997. *Adverbs and Modality in English*. New York: Addison Wesley Longman Inc.

Izard, Carroll E. 1971. *The Face of Emotion*. New York: Appleton-Century-Crofts.

Kipper, Karin, Anna Korhonen, Neville Ryant, and Martha Palmer. 2007. A Large-scale Classification of English Verbs. *Language Resources and Evaluation*, 42 (1): 21-40.

Kozareva, Zornitsa, Borja Navarro, Sonia Vazquez, and Andres Montoyo, A. 2007. UA-ZBSA: A Headline Emotion Classification through Web Information. *Proceedings of the SemEval-2007 International Workshop*, pp. 334-337.

Liu, Hugo, Henry Lieberman, and Ted Selker. 2003. A Model of Textual Affect Sensing Using Real-World Knowledge. *Proceedings of IUI-2003*, pp. 125-132.

Martin, James R., and Peter R.R. White. 2005. *The Language of Evaluation: Appraisal in English*. Palgrave, London, UK.

Miller, George A. 1990. WordNet: An On-line Lexical Database. *International Journal of Lexicography*, Special Issue, 3 (4): 235-312.

Moilanen, Karo, and Stephen Pulman. 2007. Sentiment Composition. *Proceedings of the Recent Advances in Natural Language Processing International Conference*, pp. 378-382.

Nasukawa, Tetsuya, and Jeonghee Yi. 2003. Sentiment Analysis: Capturing Favorability using Natural Language Processing. *Proceedings of the 2nd International Conference on Knowledge Capture*, pp. 70-77.

Neviarouskaya, Alena, Helmut Prendinger, and Mitsuru Ishizuka. 2009. SentiFul: Generating a Reliable Lexicon for Sentiment Analysis. *Proceedings of the International Conference on Affective Computing and Intelligent Interaction*, IEEE, Amsterdam, Netherlands, pp. 363-368.

Strapparava, Carlo, and Rada Mihalcea. 2008. Learning to Identify Emotions in Text. *Proceedings of the 2008 ACM Symposium on Applied Computing*, Fortaleza, Brazil, pp. 1556-1560.

Strapparava, Carlo, Alessandro Valitutti, and Oliviero Stock. 2007. Dances with Words. *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 1719-1724.

Subrahmanian, V.S., and Diego Reforgiato. 2008. AVA: Adjective-Verb-Adverb Combinations for Sentiment Analysis. *Intelligent Systems*, IEEE, 23 (4): 43-50.

Taboada, Maite, and Jack Grieve. 2004. Analyzing Appraisal Automatically. *Proceedings of AAAI Spring Symposium on Exploring Attitude and Affect in Text*, pp.158-161.

Whitelaw, Casey, Navendu Garg, and Shlomo Argamon. 2005. Using Appraisal Groups for Sentiment Analysis. *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, CIKM, Bremen, Germany, pp. 625-631.

Wilson, Theresa, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing Contextual Polarity in Phrase-level Sentiment Analysis. *Proceedings of HLT-EMNLP-2005*, ACL, pp. 347-354.

Nonparametric Word Segmentation for Machine Translation

ThuyLinh Nguyen Stephan Vogel Noah A. Smith
Language Technologies Institute
Carnegie Mellon University
{thuylinh,vogel,nasmith}@cs.cmu.edu

Abstract

We present an unsupervised word segmentation model for machine translation. The model uses existing monolingual segmentation techniques and models the joint distribution over source sentence segmentations and alignments to the target sentence. During inference, the monolingual segmentation model and the bilingual word alignment model are coupled so that the alignments to the target sentence guide the segmentation of the source sentence. The experiments show improvements on Arabic-English and Chinese-English translation tasks.

1 Introduction

In statistical machine translation, the smallest unit is usually the *word*, defined as a token delimited by spaces. Given a parallel corpus of source and target text, the training procedure first builds a word alignment, then extracts phrase pairs from this word alignment. However, in some languages (e.g., Chinese) there are no spaces between words.

The same problem arises when translating between two very different languages, such as from a language with rich morphology like Hungarian or Arabic to a language with poor morphology like English or Chinese. A single word in a morphologically rich language is often the composition of several morphemes, which correspond to separate words in English.[1]

Often some preprocessing is applied involving word segmentation or morphological analysis of the source and/or target text. Such preprocessing tokenizes the text into morphemes or words, which linguists consider the smallest meaning-bearing units of the language. Take as an example the Arabic word "fktbwha" and its English translation "so they wrote it". The preferred segmentation of "fktbwha" would be "f-ktb-w-ha (*so-wrote-they-it*)," which would allow for a one-to-one mapping between tokens in the two languages. However, the translation of the phrase in Hebrew is "wktbw ath". Now the best segmentation of the Arabic words would be "fktbw-ha," corresponding to the two Hebrew words. This example shows that there may not be one correct segmentation that can be established in a preprocessing step. Rather, tokenization depends on the language we want to translate into and needs to be tied in with the alignment process. In short, we want to find the tokenization yielding the best alignment, and thereby the best translation system.

We propose an unsupervised tokenization method for machine translation by formulating a generative Bayesian model to "explain" the bilingual training data. Generation of a sentence pair is described as follows: first a monolingual tokenization model generates the source sentence, then the alignment model generates the target sentence through the alignments with the source sentence. Breaking this generation process into two steps provides flexibility to incorporate existing monolingual morphological segmentation models such as those of Mochihashi et al. (2009) or Creutz and Lagus (2007). Using nonparametric

[1] We will use the terms *word segmentation*, *morphological analysis*, and *tokenization* more or less interchangeably.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 815–823,
Beijing, August 2010

models and the Bayesian framework makes it possible to incorporate linguistic knowledge as prior distributions and obtain the posterior distribution through inference techniques such as MCMC or variational inference.

As new test source sentences do not have translations which can help to infer the best segmentation, we *decode* the source string according to the posterior distribution from the inference step.

In summary, our segmentation technique consists of the following steps:

- A joint model of segmented source text and its target translation.
- Inference of the posterior distribution of the model given the training data.
- A decoding algorithm for segmenting source text.
- Experiments in translation using the preprocessed source text.

Our experiments show that the proposed segmentation method leads to improvements on Arabic-English and Chinese-English translation tasks.

In the next section we will discuss related work. Section 3 will describe our model in detail. The inference will be covered in Section 4, and decoding in Section 5. Experiments and results will be presented in Section 6.

2 Related Work

The problem of segmentation for machine translation has been studied extensively in recent literature. Most of the work used some linguistic knowledge about the source and the target languages (Nießen and Ney, 2004; Goldwater and McClosky, 2005). Sadat and Habash (2006) experimented with a wide range of tokenization schemes for Arabic-English translation. These experiments further show that even for a single language pair, different tokenizations are needed depending on the training corpus size. The experiments are very expensive to conduct and do not generalize to other language pairs. Recently, Dyer (2009) created manually crafted lattices for a subset of source words as references for segmentation when translating into English, and then learned the segmentation of the source words to optimize the translation with respect to these references. He showed that the parameters of the model can be applied to similar languages when translating into English. However, manually creating these lattices is time-consuming and requires a bilingual person with some knowledge of the underlying statistical machine translation system.

There have been some attempts to apply unsupervised methods for tokenization in machine translation (Chung and Gildea, 2009; Xu et al., 2008). The alignment model of Chung and Gildea (2009) forces every source word to align with a target word. Xu et al. (2008) modeled the source-to-null alignment as in the source word to target word model. Their models are special cases of our proposed model when the source model[2] is a unigram model. Like Xu et al. (2008), we use Gibbs sampling for inference. Chung and Gildea (2009) applied efficient dynamic programming-based variational inference algorithms.

We benefit from existing unsupervised monolingual segmentation. The source model uses the nested Pitman-Yor model as described by Mochihashi et al. (2009). When sampling each potential word boundary, our inference technique is a bilingual extension of what is described by Goldwater et al. (2006) for monolingual segmentation.

Nonparametric models have received attention in machine translation recently. For example, DeNero et al. (2008) proposed a hierarchical Dirichlet process model to learn the weights of phrase pairs to address the degeneration in phrase extraction. Teh (2006) used a hierarchical Pitman-Yor process as a smoothing method for language models.

Recent work on multilingual language learning successfully used nonparametric models for language induction tasks such as grammar induction (Snyder et al., 2009; Cohen et al., 2010), morphological segmentation (Goldwater et al., 2006; Snyder and Barzilay, 2008), and part-of-speech tagging (Goldwater and Griffiths, 2007; Snyder et al.,

[2]Note that "source model" here means a model of source text, not a source model in the noisy channel paradigm.

2008).

3 Models

We start with the generative process for a source sentence and its alignment with a target sentence. Then we describe individual models employed by this generation scheme.

3.1 Generative Story

A source sentence is a sequence of word tokens, and each word is either aligned or not aligned. We focus only on the segmentation problem and not reordering source words; therefore, the model will not generate the order of the target word tokens. A sentence pair and its alignment are captured by four components:

- a sequence of words in the source sentence,
- a set of null-aligned source tokens,
- a set of null-aligned target tokens, and
- a set of (source word to target word) alignment pairs.

We will start with a high-level story of how the segmentation of the source sentence and the alignment are generated.

1. A source language monolingual segmentation model generates the source sentence.
2. Generate alignments:
   (a) Given the sequence of words of the source sentence already generated in step 1, the alignment model marks each source word as either aligned or unaligned. If a source word is aligned, the model also generates the target word.
   (b) Unaligned target words are generated.

The model defines the joint probability of a segmented source language sentence and its alignment. During inference, the two parts are coupled, so that the alignment will influence which segmentation is selected. However, there are several advantages in breaking the generation process into two steps.

First of all, in principle the model can incorporate any existing probabilistic monolingual segmentation to generate the source sentence. For example, the source model can be the nested Pitman-Yor process as described by Mochihashi et al. (2009), the minimum description length model presented by Creutz and Lagus (2007), or something else. Also the source model can incorporate linguistic knowledge from a rule-based or statistical morphological disambiguator.

The model generates the alignment after the source sentence with word boundaries already generated. Therefore, the alignment model can be any existing word alignment model (Brown et al., 1993; Vogel et al., 1996). Even though the choices of source model or alignment model can lead to different inference methods, the model we propose here is highly extensible. Note that we assume that the alignment consists of at most one-to-one mappings between source and target words, with null alignments possible on both sides.

Another advantage of a separate source model lies in the segmentation of an unseen test set. In section 5 we will show how to apply the source model distribution learned from training data to find the best segmentation of an unseen test set.

Notation and Parameters

We will use bold font for a sequence or bags of words and regular font for an individual word. A source sentence $\mathbf{s}$ is a sequence of $|\mathbf{s}|$ words s_i: $\left(s_1, \ldots, s_{|\mathbf{s}|}\right)$; the translation of sentence $\mathbf{s}$ is the target sentence $\mathbf{t}$ of $|\mathbf{t}|$ words $\left(t_1, \ldots, t_{|\mathbf{t}|}\right)$. In sentence $\mathbf{s}$ the list of unaligned words is $\mathbf{s}_{\text{nal}}$ and the list of aligned source words is $\mathbf{s}_{\text{al}}$. In the target sentence $\mathbf{t}$ the list of unaligned words is $\mathbf{t}_{\text{nal}}$ and the list of target words having one-to-one alignment with source words $\mathbf{s}_{\text{al}}$ is $\mathbf{t}_{\text{al}}$. The alignment $\mathbf{a}$ of $\mathbf{s}$ and $\mathbf{t}$ is represented by $\{\langle s_i, \mathsf{null}\rangle \mid s_i \in \mathbf{s}_{\text{nal}}\} \cup \{\langle s_i, t_{a_i}\rangle \mid s_i \in \mathbf{s}_{\text{al}}; t_{a_i} \in \mathbf{t}_{\text{al}}\} \cup \{\langle \mathsf{null}, t_j\rangle \mid t_j \in \mathbf{t}_{\text{nal}}\}$ where a_i denotes the index in $\mathbf{t}$ of the word aligned to s_i.

The probability of a sequence or a set is denoted by $\mathbb{P}(.)$, probability at the word level is $\mathsf{p}(.)$. For example, the probability of sentence $\mathbf{s}$ is $\mathbb{P}(\mathbf{s})$, the probability of a word s is $\mathsf{p}(s)$, the probability that the target word t aligns to an aligned source

word s is $\mathsf{p}(t \mid s)$.

A sentence pair and its alignment are generated from the following models:

- The *source* model generates sentence $\mathbf{s}$ with probability $\mathbb{P}(\mathbf{s})$.
- The *source-to-null* alignment model decides independently for each word s whether it is unaligned with probability $\mathsf{p}(\mathsf{null} \mid s_i)$ or aligned with probability: $1 - \mathsf{p}(\mathsf{null} \mid s_i)$. The probability of this step, for all source words, is: $\mathbb{P}(\mathbf{s}_{\text{nal}}, \mathbf{s}_{\text{al}} \mid \mathbf{s}) = \prod_{s_i \in \mathbf{s}_{\text{nal}}} \mathsf{p}(\mathsf{null} \mid s_i) \times \prod_{s_i \in \mathbf{s}_{\text{al}}} (1 - \mathsf{p}(\mathsf{null} \mid s_i))$.

  We will also refer to the source-to-null model as the *deletion model*, since words in $\mathbf{s}_{\text{nal}}$ are effectively deleted for the purposes of alignment.
- The *source-to-target* alignment model generates a bag of target words $\mathbf{t}_{\text{al}}$ aligned to the source words $\mathbf{s}_{\text{al}}$ with probability: $\mathbb{P}(\mathbf{t}_{\text{al}} \mid \mathbf{s}_{\text{al}}) = \prod_{s_i \in \mathbf{s}_{\text{al}}; t_{a_i} \in \mathbf{t}_{\text{al}}} \mathsf{p}(t_{a_i} \mid s_i)$. Note that we do not need to be concerned with generating $\mathbf{a}$ explicitly, since we do not model word order on the target side.
- The *null-to-target* alignment model generates the list of unaligned target words $\mathbf{t}_{\text{nal}}$ given aligned target words $\mathbf{t}_{\text{al}}$ with $\mathbb{P}(\mathbf{t}_{\text{nal}} \mid \mathbf{t}_{\text{al}})$ as follows:
  - Generate the number of unaligned target words $|\mathbf{t}_{\text{nal}}|$ given the number of aligned target words $|\mathbf{t}_{\text{al}}|$ with probability $\mathbb{P}(|\mathbf{t}_{\text{nal}}| \mid |\mathbf{t}_{\text{al}}|)$.
  - Generate $|\mathbf{t}_{\text{nal}}|$ unaligned words $t \in \mathbf{t}_{\text{nal}}$ independently, each with probability $\mathsf{p}(t \mid \mathsf{null})$.

  The resulting null-to-target probability is therefore: $\mathbb{P}(\mathbf{t}_{\text{nal}} \mid \mathbf{t}_{\text{al}}) = \mathbb{P}(|\mathbf{t}_{\text{nal}}| \mid |\mathbf{t}_{\text{al}}|) \prod_{t \in \mathbf{t}_{\text{nal}}} \mathsf{p}(t \mid \mathsf{null})$.

  We also call the null-to-target model the *insertion* model.

The above generation process defines the joint probability of source sentence $\mathbf{s}$ and its alignment $\mathbf{a}$ as follows:

$$\mathbb{P}(\mathbf{s}, \mathbf{a}) = \underbrace{\mathbb{P}(\mathbf{s})}_{\text{source model}} \times \underbrace{\mathbb{P}(\mathbf{a} \mid \mathbf{s})}_{\text{alignment model}} \quad (1)$$

$$\mathbb{P}(\mathbf{a} \mid \mathbf{s}) = \mathbb{P}(\mathbf{t}_{\text{al}} \mid \mathbf{s}_{\text{al}}) \times \mathbb{P}(\mathbf{t}_{\text{nal}} \mid \mathbf{t}_{\text{al}}) \times \prod_{s_i \in \mathbf{s}_{\text{nal}}} \mathsf{p}(\mathsf{null} \mid s_i) \times \prod_{s_i \in \mathbf{s}_{\text{al}}} (1 - \mathsf{p}(\mathsf{null} \mid s_i)) \quad (2)$$

3.2 Source Model

Our generative process provides the flexibility of incorporating different monolingual models into the probability distribution of a sentence pair. In particular we use the existing state-of-the-art nested Pitman-Yor n-gram language model as described by Mochihashi et al. (2009). The probability of $\mathbf{s}$ is given by

$$\mathbb{P}(\mathbf{s}) = \mathbb{P}(|\mathbf{s}|) \prod_{i=1}^{|\mathbf{s}|} \mathsf{p}(s_i \mid s_{i-n}, \ldots, s_{i-1}) \quad (3)$$

where the n-gram probability is a hierarchical Pitman-Yor language model using $(n-1)$-gram as the base distribution.

At the unigram level, the model uses the base distribution $\mathsf{p}(s)$ as the infinite-gram character-level Pitman-Yor language model.

3.3 Modeling Null-Aligned Source Words

The probability that a source word aligns to null $\mathsf{p}(\mathsf{null} \mid s)$ is defined by a binomial distribution with Beta prior $\mathrm{Beta}(\alpha p, \alpha(1-p))$, where α and p are model parameters. When $p \to 0$ and $\alpha \to \infty$ the probability $\mathsf{p}(\mathsf{null} \mid s)$ converges to 0 forcing each source words align to a target word. We fixed $p = 0.1$ and $\alpha = 20$ in our experiment.

Xu et al. (2008) view the null word as another target word, hence in their model the probability that a source word aligns to null can only depend on itself.

By modeling the source-to-null alignment separately, our model lets the distribution depend on the word's n-gram context as in the source model. $\mathsf{p}(\mathsf{null} \mid s_{i-n}, \ldots, s_i)$ stands for the probability that the word s_i is not aligned given its context $(s_{i-n}, \ldots, s_{i-1})$.

The n-gram source-to-null distribution $\mathsf{p}(\mathsf{null} \mid s_{i-n}, \ldots, s_i)$ is defined similarly to

$\mathsf{p}(\mathsf{null} \mid s_i)$ definition above in which the base distribution p now becomes the $(n-1)$-gram: $\mathsf{p}(\mathsf{null} \mid s_{i-n+1}, \ldots, s_i)$.[3]

3.4 Source-Target Alignment Model

The probability $\mathsf{p}(t \mid s)$ that a target word t aligns to a source word s is a Pitman-Yor process:

$$t \mid s \sim \mathrm{PY}(d, \alpha, \mathsf{p}_0(t \mid s))$$

here d and α are the input parameters, and $\mathsf{p}_0(t \mid s)$ is the base distribution.

Let $|s, \cdot|$ denote the number of times s is aligned to any t in the corpus and let $|s, t|$ denote the number of times s is aligned to t anywhere in the corpus. And let $\mathsf{ty}(s)$ denote the number of different target words t the word s is aligned to anywhere in the corpus. In the Chinese Restaurant Process metaphor, there is one restaurant for each source word s, the s restaurant has $\mathsf{ty}(s)$ tables and total $|s, \cdot|$ customers; table t has $|s, t|$ customers.

Then, at a given time in the generative process for the corpus, we can write the probability that t is generated by the word s as:

- if $|s, t| > 0$:

$$\mathsf{p}(t \mid s) = \frac{|s, t| - d + [\alpha + d\mathsf{ty}(s)]\mathsf{p}_0(t \mid s)}{|s, \cdot| + \alpha}$$

- if $|s, t| = 0$:

$$\mathsf{p}(t \mid s) = \frac{[\alpha + d\mathsf{ty}(s)]\mathsf{p}_0(t \mid s)}{|s, \cdot| + \alpha}$$

For language pairs with similar character sets such as English and French, words with similar surface form are often translations of each other. The base distribution can be defined based on the edit distance between two words (Snyder and Barzilay, 2008).

We are working with diverse language pairs (Arabic-English and Chinese-English), so we use the base distribution as the flat distribution $\mathsf{p}_0(t \mid s) = \frac{1}{T}$; T is the number of distinct target words in the training set. In our experiment, the model parameters are $\alpha = 20$ and $d = .5$.

[3] We also might have conditioned this decision on words *following* s_i, since those have all been generated already at this stage.

3.5 Modeling Null-Aligned Target Words

The null-aligned target words are modeled conditioned on previously generated target words as:

$$\mathbb{P}(\mathbf{t}_{\mathrm{nal}} \mid \mathbf{t}_{\mathrm{al}}) = \mathbb{P}(|\mathbf{t}_{\mathrm{nal}}| \mid |\mathbf{t}_{\mathrm{al}}|) \prod_{t \in \mathbf{t}_{\mathrm{nal}}} \mathsf{p}(t \mid \mathsf{null})$$

This model uses two probability distributions:

- the number of unaligned target words: $\mathbb{P}(|\mathbf{t}_{\mathrm{nal}}| \mid |\mathbf{t}_{\mathrm{al}}|)$, and
- the probability that each word in $\mathbf{t}_{\mathrm{nal}}$ is generated by null: $\mathsf{p}(t \mid \mathsf{null})$.

We model the number of unaligned target words similarly to the distribution in the IBM3 word alignment model (Brown et al., 1993). IBM3 assumes that each aligned target words generates a null-aligned target word with probability p_0 and fails to generate a target word with probability $1 - p_0$. So the parameter p_0 can be used to control the number of unaligned target words. In our experiments, we fix $p_0 = .05$. Following this assumption, the probability of $|\mathbf{t}_{\mathrm{nal}}|$ unaligned target words generated from $|\mathbf{t}_{\mathrm{al}}|$ words is: $\mathbb{P}(|\mathbf{t}_{\mathrm{nal}}| \mid |\mathbf{t}_{\mathrm{al}}|) = \binom{|\mathbf{t}_{\mathrm{al}}|}{|\mathbf{t}_{\mathrm{nal}}|} p_0^{|\mathbf{t}_{\mathrm{nal}}|}(1 - p_0)^{|\mathbf{t}_{\mathrm{al}}| - |\mathbf{t}_{\mathrm{nal}}|}$.

The probability that a target word t aligns to null, $\mathsf{p}(t \mid \mathsf{null})$, also has a Pitman-Yor process prior. The base distribution of the model is similar to the source-to-target model's base distribution which is the flat distribution over target words.

4 Inference

We have defined a probabilistic generative model to describe how a corpus of alignments and segmentations can be generated jointly. In this section we discuss how to obtain the posterior distributions of the missing alignments and segmentations given the training corpus, using Gibbs sampling.

Suppose we are provided a morphological disambiguator for the source language such as MADA morphology tokenization toolkit (Sadat and Habash, 2006) for Arabic.[4] The morphological disambiguator segments a source word to

[4] MADA provides several segmentation schemes; among them the MADA-D3 scheme seeks to separate all morphemes of each word.

morphemes of smallest meaning-bearing units of the source language. Therefore, a target word is equivalent to one or several morphemes. Given a morphological disambiguation toolkit, we use its output to bias our inference by not considering word boundaries after every character but only considering potential word boundaries as a subset of the morpheme boundaries set. In this way, the inference uses the morphological disambiguation toolkit to limit its search space.

The inference starts with an initial segmentation of the source corpus and also its alignment to the target corpus. The Gibbs sampler considers one potential word boundary at a time. There are two hypotheses at any given boundary position of a sentence pair $(\mathbf{s}, \mathbf{t})$: the *merge* hypothesis stands for no word boundary and the resulting source sentence $\mathbf{s}_{\text{merge}}$ has a word s spanning over the sample point; the *split* hypothesis indicates the resulting source sentence $\mathbf{s}_{\text{split}}$ has a word boundary at the sample point separating two words $s_1 s_2$. Similar to Goldwater et al. (2006) for monolingual segmentation, the sampler randomly chooses the boundary according to the relative probabilities of the merge hypothesis and the split hypothesis.

The model consists of source and alignment model variables; given the training corpora size of a machine translation system, the number of variables is large. So if the Gibbs sampler samples both source variables and alignment variables, the inference requires many iterations until the sampler mixes. Xu et al. (2008) fixed this by repeatedly applying GIZA++ word alignment after each sampling iteration through the training corpora.

Our inference technique is not precisely Gibbs sampling. Rather than sampling the alignment or attempting to collapse it out (by summing over all possible alignments when calculating the relative probabilities of the merge and split hypotheses), we seek the *best* alignment for each hypothesis. In other words, for each hypothesis, we perform a local search for a high-probability alignment of the merged word or split words, given the rest of alignment for the sentence. Up to one word may be displaced and realigned. This "local-best" alignment is used to score the hypothesis, and after sampling merge or split, we keep that best alignment.

This inference technique is motivated by runtime demands, but we do not yet know of a theoretical justification for combining random steps with maximization over some variables. A more complete analysis is left to future work.

5 Decoding for Unseen Test Sentences

Section 4 described how to get the model's posterior distribution and the segmentation and alignment of the training data under the model. We are left with the problem of *decoding* or finding the segmentation of test sentences where the translations are not available. This is needed when we want to translate new sentences. Here, tokenization is performed as a preprocessing step, decoupled from the subsequent translation steps.

The decoding step uses the model's posterior distribution for the training data to segment unseen source sentences. Because of the clear separation of the source model and the alignment model, the source model distribution learned from the Gibbs sampling directly represents the distribution over the source language and can therefore also handle the segmentation of unknown words in new test sentences. Only the source model is used in preprocessing.

The best segmentation $\mathbf{s}^*$ of a string of characters $\mathbf{c} = (c_1, \ldots, c_{|\mathbf{c}|})$ according to the n-gram source model is:

$$\mathbf{s}^* = \underset{\mathbf{s} \text{ from } \mathbf{c}}{\operatorname{argmax}} \, \mathsf{p}(|\mathbf{s}|) \prod_{i=1}^{i=|\mathbf{s}|} \mathsf{p}(s_i \mid s_{i-n}, \ldots, s_{i-1})$$

We use a stochastic finite-state machine for decoding. This is possible by composition of the following two finite state machines:

- Acceptor $\mathcal{A}_{\mathbf{c}}$. The string of characters $\mathbf{c}$ is represented as an finite state acceptor machine where any path through the machine represents an unweighted segmentation of $\mathbf{c}$.
- Source model weighted finite state transducer $\mathcal{L}_{\mathbf{c}}$. Knight and Al-Onaizan (1998) show how to build an n-gram language model by a weighted finite state machine. The states of the transducer are $(n-1)$-gram history, the edges are words from the language. The arc s_i coming from state

$(s_{i-n}, \ldots, s_{i-1})$ to state $(s_{i-n+1}, \ldots, s_i)$ has weight $\mathsf{p}(s_i \mid s_{i-n}, \ldots, s_{i-1})$.

The best segmentation $\mathbf{s}^*$ is given as $\mathbf{s}^* = \mathrm{BestPath}(\mathcal{A}_\mathbf{c} \circ \mathcal{L}_\mathbf{c})$.

6 Experiments

This section presents experimental results on Arabic-English and Chinese-English translation tasks using the proposed segmentation technique.

6.1 Arabic-English

As a training set we use the BTEC corpus distributed by the International Workshop on Spoken Language Translation (IWSLT) (Matthias and Chiori, 2005). The corpus is a collection of conversation transcripts from the travel domain. The "Supplied Data" track consists of nearly 20K Arabic-English sentence pairs. The development set consists of 506 sentences from the IWSLT04 evaluation test set and the unseen set consists of 500 sentences from the IWSLT05 evaluation test set. Both development set and test set have 16 references per Arabic sentence.

6.2 Chinese-English

The training set for Chinese-English translation task is also distributed by the IWSLT evaluation campaign. It consists of 67K Chinese-English sentence pairs. The development set and the test set each have 489 Chinese sentences and each sentence has 7 English references.

6.3 Results

We will report the translation results where the preprocessing of the source text are our unigram, bigram, and trigram source models and source-to-null model.

The MCMC inference algorithm starts with an initial segmentation of the source text into full word forms. For Chinese, we use the original word segmentation as distributed by IWSLT. To get an initial alignment, we generate the IBM4 Viterbi alignments in both directions using the GIZA++ toolkit (Och and Ney, 2003) and combine them using the "grow-diag-final-and" heuristic. The output of combining GIZA++ alignment for a sentence pair is a sequence of s_i-t_j entries where i is an index of the source sentence and j is an index of the target sentence. As our model allows only one-to-one mappings between the words in the source and target sentences, we remove s_i-t_j from the sequence if either the source word s_i or target word t_j is already in a previous entry of the combined alignment sequence. The resulting alignment is our initial alignment for the inference.

We also apply the MADA morphology segmentation toolkit (Habash and Rambow, 2005) to preprocess the Arabic corpus. We use the D3 scheme (each Arabic word is segmented into morphemes in sequence [CONJ+ [PART+ [Al+ BASE +PRON]]]), mark the morpheme boundaries, and then combine the morphemes again to have words in their original full word form. During inference, we only sample over these morpheme boundaries as potential word boundaries. In this way, we limit the search space, allowing only segmentations consistent with MADA-D3.

The inference samples 150 iterations through the whole training set and uses the posterior probability distribution from the last iteration for decoding. The decoding process is then applied to the entire training set as well as to the development and test sets to generate a consistent tokenization across all three data sets. We used the OpenFST toolkit (Allauzen et al., 2007) for finite-state machine implementation and operations. The output of the decoding is the preprocessed data for translation. We use the open source Moses phrase-based MT system (Koehn et al., 2007) to test the impact of the preprocessing technique on translation quality.[5]

6.3.1 Arabic-English Translation Results

We consider the Arabic-English setting. We use two baselines: original full word form and MADA-D3 tokenization scheme for Arabic-English translation. Table 1 compares the translation results of our segmentation methods with these baselines. Our segmentation method shows improvement over the two baselines on both the development and test sets. According to Sadat and Habash (2006), the MADA-D3 scheme per-

[5]The Moses translation alignment is the output of GIZA++, not from our MCMC inference.

	Dev.	Test
Original	59.21	54.00
MADA-D3	58.28	54.92
Unigram	**59.44**	56.18
Bigram	58.88	56.18
Trigram	58.76	**56.82**

Table 1: Arabic-English translation results (BLEU).

forms best for their Arabic-English translation especially for small and moderate data sizes. In our experiments, we see an improvement when using the MADA-D3 preprocessing over using the original Arabic corpus on the unseen test set, but not on the development set.

The Gibbs sampler only samples on the morphology boundary points of MADA-D3, so the improvement resulting from our segmentation technique does not come from removing unknown words. It is due to a better matching between the source and target sentences by integrating segmentation and alignment. We therefore expect the same impact on a larger training data set in future experiments.

6.3.2 Chinese-English Translation Results

	Dev.	Test
Whole word	23.75	**29.02**
Character	23.39	27.74
Unigram	**24.90**	**28.97**
Trigram	23.98	28.20

Table 2: Chinese-English translation result in BLEU score metric.

We next consider the Chinese-English setting. The translation performance using our word segmentation technique is shown in Table 2. There are two baselines for Chinese-English translation: (a) the source text in the full word form distributed by the IWSLT evaluation and (b) no segmentation of the source text, which is equivalent to interpreting each Chinese character as a single word.

Taking development and test sets into account, the best Chinese-English translation system results from our unigram model. It is significantly better than other systems on the development set and performs almost equally well with the IWSLT segmentation on the test set. Note that the segmentation distributed by IWSLT is a manual segmentation for the translation task.

Chung and Gildea (2009) and Xu et al. (2008) also showed improvement over a simple monolingual segmentation for Chinese-English translation. Our character-based translation result is comparable to their monolingual segmentations. Both trigram and unigram translation results outperform the character-based translation.

We also observe that there are no additional gains for Chinese-English translation when using a higher n-gram model. Our Gibbs sampler has the advantage that the samples are guaranteed to converge eventually to the model's posterior distributions, but in each step the modification to the current hypothesis is small and local. In iterations 100–150, the average number of boundary changes for the unigram model is 14K boundaries versus only 1.5K boundary changes for the trigram model. With 150 iterations, the inference output of trigram model might not yet represent its posterior distribution. We leave a more detailed investigation of convergence behavior to future work.

Conclusion and Future Work

We presented an unsupervised segmentation method for machine translation and presented experiments for Arabic-English and Chinese-English translation tasks. The model can incorporate existing monolingual segmentation models and seeks to learn a segmenter appropriate for a particular translation task (target language and dataset).

Acknowledgements

We thank Kevin Gimpel for interesting discussions and technical advice. We also thank the anonymous reviewers for useful feedback. This work was supported by DARPA Gale project, NSF grants 0844507 and 0915187.

References

Allauzen, C., M. Riley, J. Schalkwyk, W. Skut, and M. Mohri. 2007. OpenFst: A General and Efficient Weighted Finite-State Transducer Library. In *Proceedings of the CIAA 2007*, volume 4783 of *Lecture Notes in Computer Science*, pages 11–23. Springer. `http://www.openfst.org`.

Brown, Peter F., Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Comput. Linguist.*, 19(2):263–311.

Chung, T. and D. Gildea. 2009. Unsupervised Tokenization for Machine Translation. In *Proceedings of EMNLP 2009*, pages 718–726, Singapore, August. Association for Computational Linguistics.

Cohen, S. B., D. M. Blei, and N. A. Smith. 2010. Variational Inference for Adaptor Grammars. In *Proceedings of NAACL-HLT*, pages 564–572, June.

Creutz, Mathias and Krista Lagus. 2007. Unsupervised Models for Morpheme Segmentation and Morphology Learning. *ACM Trans. Speech Lang. Process.*, 4(1):1–34.

DeNero, J., A. Bouchard-Côté, and D. Klein. 2008. Sampling Alignment Structure under a Bayesian Translation Model. In *Proceedings of EMNLP 2008*, pages 314–323, Honolulu, Hawaii, October. Association for Computational Linguistics.

Dyer, C. 2009. Using a Maximum Entropy model to build segmentation lattices for MT. In *Proceedings of HLT 2009*, pages 406–414, Boulder, Colorado, June.

Goldwater, S. and T. L. Griffiths. 2007. A Fully Bayesian Approach to Unsupervised Part-of-Speech Tagging. In *Proceedings of ACL*.

Goldwater, S. and D. McClosky. 2005. Improving Statistical Machine Translation Through Morphological Analysis. In *Proc. of EMNLP*.

Goldwater, S., T. L. Griffiths, and M. Johnson. 2006. Contextual Dependencies in Unsupervised Word Segmentation. In *Proc. of COLING-ACL*.

Habash, N. and O. Rambow. 2005. Arabic Tokenization, Part-of-Speech Tagging, and Morphological Disambiguation in One Fell Swoop. In *Proc. of ACL*.

Knight, K. and Y. Al-Onaizan. 1998. Translation with Finite-State Devices. In *Proceedings of AMTA*, pages 421–437.

Koehn, P., H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proc. of ACL (demo session)*.

Matthias, E. and H. Chiori. 2005. Overview of the IWSLT 2005 Evaluation Campaign. In *Proceedings of IWSLT*.

Mochihashi, D., T. Yamada, and N. Ueda. 2009. Bayesian Unsupervised Word Segmentation with Nested Pitman-Yor Language Modeling. In *Proceedings of 47th ACL*, pages 100–108, Suntec, Singapore, August.

Nießen, S. and H. Ney. 2004. Statistical Machine Translation with Scarce Resources Using Morpho-Syntactic Information. *Computational Linguistics*, 30(2), June.

Och, F. and H. Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1).

Sadat, F. and N. Habash. 2006. Combination of Arabic Preprocessing Schemes for Statistical Machine Translation. In *Proceedings of the ACL*, pages 1–8.

Snyder, B. and R. Barzilay. 2008. Unsupervised Multilingual Learning for Morphological Segmentation. In *Proceedings of ACL-08: HLT*, pages 737–745, June.

Snyder, B., T. Naseem, J. Eisenstein, and R. Barzilay. 2008. Unsupervised Multilingual Learning for POS Tagging. In *Proceedings of EMNLP*.

Snyder, B., T. Naseem, and R. Barzilay. 2009. Unsupervised Multilingual Grammar Induction. In *Proceedings of ACL-09*, pages 73–81, August.

Teh, Y. W. 2006. A Hierarchical Bayesian Language Model Based On Pitman-Yor Processes. In *Proceedings of ACL*, pages 985–992, July.

Vogel, S., H. Ney, and C. Tillmann. 1996. HMM-Based Word Alignment in Statistical Translation. In *Proceedings of COLING*, pages 836–841.

Xu, J., J. Gao, K. Toutanova, and H. Ney. 2008. Bayesian Semi-Supervised Chinese Word Segmentation for Statistical Machine Translation. In *Proceedings of (Coling 2008)*, pages 1017–1024, Manchester, UK, August.

Automatic Discovery of Feature Sets for Dependency Parsing

Peter Nilsson **Pierre Nugues**
Department of Computer Science
Lund University
peter.nilsson.lund@telia.com Pierre.Nugues@cs.lth.se

Abstract

This paper describes a search procedure to discover optimal feature sets for dependency parsers. The search applies to the shift–reduce algorithm and the feature sets are extracted from the parser configuration. The initial feature is limited to the first word in the input queue. Then, the procedure uses a set of rules founded on the assumption that topological neighbors of significant features in the dependency graph may also have a significant contribution. The search can be fully automated and the level of greediness adjusted with the number of features examined at each iteration of the discovery procedure.

Using our automated feature discovery on two corpora, the Swedish corpus in CoNLL-X and the English corpus in CoNLL 2008, and a single parser system, we could reach results comparable or better than the best scores reported in these evaluations. The CoNLL 2008 test set contains, in addition to a *Wall Street Journal* (WSJ) section, an out-of-domain sample from the Brown corpus. With sets of 15 features, we obtained a labeled attachment score of 84.21 for Swedish, 88.11 on the WSJ test set, and 81.33 on the Brown test set.

1 Introduction

The selection of relevant feature sets is crucial to the performance of dependency parsers and this process is still in large part manual. Moreover, feature sets are specific to the languages being analyzed and a set optimal for, say, English can yield poor results in Chinese. With dependency parsers being applied today to dozens of languages, this makes the parametrization of a parser both a tedious and time-consuming operation. Incidentally, the advent of machine-learning methods seems to have shifted the tuning steps in parsing from polishing up grammar rules to the optimization of feature sets. And as with the writing of a grammar, the selection of features is a challenging task that often requires a good deal of effort and inspiration.

Most automatic procedures to build feature sets resort to greedy algorithms. Forward selection constructs a set by adding incrementally features from a predetermined superset while backward elimination removes them from the superset (Attardi et al., 2007). Both methods are sometimes combined (Nivre et al., 2006b). The selection procedures evaluate the relevance of a candidate feature in a set by its impact on the overall parsing score: Does this candidate improve or decrease the performance of the set?

Greedy search, although it simplifies the design of feature sets, shows a major drawback as it starts from a closed superset of what are believed to be the relevant features. There is a broad consensus on a common feature set including the words close to the top of the stack or the beginning of the queue, for the shift–reduce algorithm, but no clear idea on the limits of this set.

In this paper, we describe an automatic discovery procedure that is not bounded by any prior knowledge of a set of potentially relevant features. It applies to the shift–reduce algorithm and the ini-

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 824–832,
Beijing, August 2010

tial feature consists solely of the first word of the queue. The search explores nodes along axes of the parser's data structures and the partially built graph using proximity rules to uncover sequences of relevant, efficient features. Using this procedure on the Swedish corpus in CoNLL-X and the English corpus in CoNLL 2008, we built feature sets that enabled us to reach a labeled attachment score of 84.21 for Swedish, 88.11 on the *Wall Street Journal* section of CoNLL 2008, and 81.33 on the Brown part of it with a set cardinality of 15.

2 Transition-based Parsing

Transition-based methods (Covington, 2001; Nivre, 2003; Yamada and Matsumoto, 2003; Zhang and Clark, 2009) have become a popular approach in multilingual dependency parsing because of their speed and performance. Transition-based methods share common properties and build a dependency graph from a sequence of actions, where each action is determined using a feature function. In a data-driven context, the function is typically implemented as a classifier and the features are extracted from the partially built graph and the parser's data structures, most often a queue and a stack.

2.1 Parser Implementation

In this study, we built a parser using Nivre's algorithm (Nivre, 2003). The parser complexity is linear and parsing completes in at most $2n+1$ operations, where n is the length of the sentence. Table 1 shows the transitions and actions to construct a dependency graph.

Given a sentence to parse, we used a classifier-based guide to predict the transition sequence to apply. At each step, the guide extracts features from the parser configuration and uses them as input to a classifier to predict the next transition. Before training the classification models, we projectivized the corpus sentences (Kunze, 1967; Nivre and Nilsson, 2005). We did not attempt to recover nonprojective sentences after parsing.

2.2 Training and Parsing Procedure

We extracted the features using a gold-standard parsing of the training set. We organized the classification, and hence the feature extraction, as a two-step process. The first step determines the action among *LeftArc*, *RightArc*, *Reduce*, and *Shift*; the second one, the grammatical function, if the action is either a left arc or a right arc.

Once the features are extracted, we train the corresponding models that we apply to the test corpus to predict the actions and the arc labels.

Action	Parser configuration
Init.	$\langle nil, W, \emptyset \rangle$
End	$\langle S, nil, G \rangle$
LeftArc	$\langle n\|S, n'\|Q, G \rangle \rightarrow$ $\langle S, n'\|Q, G \cup \{\langle n', n \rangle\} \rangle$
RightArc	$\langle n\|S, n'\|Q, G \rangle \rightarrow$ $\langle n'\|n\|S, Q, G \cup \{\langle n, n' \rangle\} \rangle$
Reduce	$\langle n\|S, Q, G \rangle \rightarrow \langle S, Q, G \rangle$
Shift	$\langle S, n\|Q, G \rangle \rightarrow \langle n\|S, Q, G \rangle$

Table 1: Parser transitions (Nivre, 2003). W is the input, G, the graph, S, the stack, and Q, the queue. The triple $\langle S, Q, G \rangle$ represents the parser configuration and n, n', and n'' are lexical tokens. $\langle n', n \rangle$ represents an arc from n' to n.

3 Feature Discovery

We designed an automatic procedure to discover and select features that is guided by the structure of the graph being constructed. The search algorithm is based on the assumption that if a feature makes a significant contribution to the parsing performance, then one or more of its topological neighbors in the dependency graph may also be significant. The initial state, from which we derive the initial feature, consists of the first word in the queue. There is no other prior knowledge on the features.

3.1 Node Attributes

In the discovery procedure, we considered the nodes of four data structures: the queue, the stack, the sentence, and the graph being constructed. We extracted three attributes (or fields) from each node: two static ones, the lexical value of the node and its part of speech, and a dynamic one evaluated at parse time: the dependency label of the arc linking the node to its head, if it exists. We denoted the attributes of node w, respectively,

$LEX(w)$, $POS(w)$, and $DEP(w)$. These attributes are used as input by most dependency parsers, whatever the language being parsed.

3.2 Search Axes

The feature search covers three different axes: the parser's data structures – the queue and the stack –, the graph being constructed, and the sentence. Given a feature set at step *n* of the discovery procedure, we defined a successor function that generates the set of topological neighbors of all the members in the feature set along these three axes. For a particular feature:

The data structure axis consists of the nodes in the stack and the queue. The immediate neighbors of a node in the stack are the adjacent nodes above and below. In the queue, these are the adjacent nodes before and after it. The top node on the stack and the next node in the queue have a special connection, since they are the ones used by the parser when creating an arc. Therefore, we considered them as immediate neighbors to each other. For a node that is neither in the stack, nor in the queue, there is no connection along this axis.

The graph axes traverse the partially constructed graph horizontally and vertically. The horizontal axis corresponds to the sibling nodes connected by a common head (Figure 1). The immediate neighbors of a node are its nearest siblings to the left and to the right. The vertical axis corresponds to the head and child nodes. The immediate neighbors are the head node as well as the leftmost and rightmost child nodes. There is no connection for nodes not yet part of the graph.

The sentence axis traverses the nodes in the order they occur in the original sentence. The immediate neighbors of a node are the previous and next words in the sentence.

4 Representing Features and Their Neighbors

We represented features with a parameter format partly inspired by MaltParser (Nivre et al., 2006a).

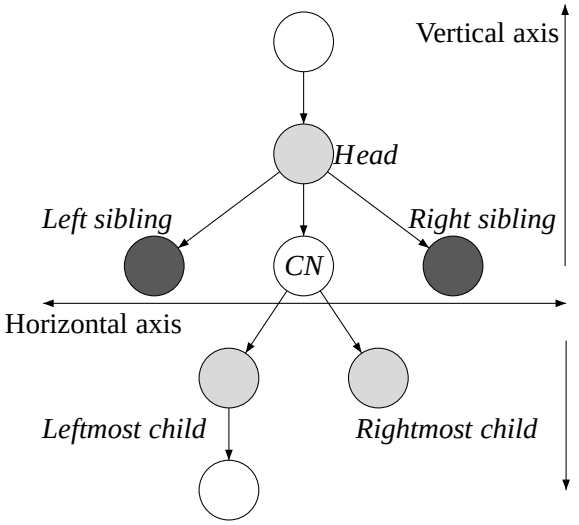


Figure 1: The vertical and horizontal axes, respectively in light and dark gray, relative to *CN*.

Each parameter consists of two parts. The first one represents a node in a data structure (*STACK* or *QUEUE*) and an attribute:

The nodes are identified using a zero-based index. Thus $STACK_1$ designates the second node on the stack.

The attribute of a node is one of part of speech (*POS*), lexical value (*LEX*), or dependency label (*DEP*), as for instance $LEX(QUEUE_0)$ that corresponds to the lexical value of the first token in the queue.

The second part of the parameter is an optional navigation path that allows to find other destination nodes in the graph. It consists of a sequence of instructions to move from the start node to the destination node. The list of possible instructions are:

- *h*: head of the current node;
- lc/rc: leftmost/rightmost child of the node;
- pw/fw: previous/following word of the node in the original sentence.

An example of a feature obtained using the navigation part is $POS(STACK_1\ lc\ pw)$, which is interpreted as: start from $STACK_1$. Then, using the instructions *lc* and *pw*, move to the left child of the start node and to the previous word of this child in the sentence. The requested feature is the part of speech of the destination node.

5 Initial State and Successor Function

The feature discovery is an iterative procedure that grows the feature set with one new feature at each iteration. We called *generation* such an iteration, where generation 1 consists of a single node. We denoted $FeatSet_i = \{f_1, f_2, ..., f_i\}$ the feature set obtained at generation i.

Although the features of a classifier can be viewed as a set, we also considered them as a tuple, where $Feat_i = \langle f_1, f_2, ..., f_i \rangle$ is the i-tuple at generation i and f_k, the individual feature discovered at generation k with $1 \leqslant k \leqslant i$. This enables us to keep the order in which the individual features are obtained during the search.

5.1 Initial State

We start the feature search with the empty set, $\emptyset$, that, by convention, has one neighbor: the first node in the queue $QUEUE_0$. We chose this node because this is the only one which is certain to exist all along the parsing process. Intuitively, this is also obvious that $QUEUE_0$ plays a significant role when deciding a parsing action. We defined the successor function of the empty set as: $SUCC(\emptyset) = \{POS(QUEUE_0), LEX(QUEUE_0)\}$.

5.2 Successors of a Node

The successors of a node consist of itself and all its topological neighbors along the three axes with their three possible attributes: part of speech, lexical value, and dependency label. For a particular feature in *FeatSet*, the generation of its successors is carried out through the following steps:

1. Interpret the feature with its possible navigation path and identify the destination node n.
2. Find all existing immediate neighboring nodes of n along the three search axes.
3. Assign the set of attributes – *POS*, *LEX*, and *DEP* – to n and its neighboring nodes.

If at any step the requested node does not exist, the feature evaluates to *NOTHING*.

5.3 Rules to Generate Neighbors

The generation of all the neighbors of the features in *FeatSet* may create duplicates as a same node can sometimes be reached from multiple paths. For instance, if we move to the leftmost child of a node and then to the head of this child, we return to the original node.

To compute the successor function, we built a set of rules shown in Table 2. It corresponds to a subset of the rules described in the axis search (Sect. 3.2) so that it omits the neighbors of a node that would unavoidably create redundancies. The third column in Table 2 shows the rules to generate the neighbors of $POS(QUEUE_0)$. They correspond to the rows:

PL. This stands for the *POS* and *LEX* attributes of the node. We only add $LEX(QUEUE_0)$ as we already have $POS(QUEUE_0)$.

PLD lc *and* PLD rc. *POS*, *LEX*, and *DEP* of the node's leftmost and rightmost children.

PLD pw. *POS*, *LEX*, and *DEP* of the previous word in the original string. The following word is the same as the next node in the queue, which is added in the next step. For that reason, *following word* is not added.

PL $QUEUE_1$**.** *POS* and *LEX* of $QUEUE_1$.

PLD $STACK_0$**.** *POS*, *LEX*, and *DEP* of $STACK_0$. This rule connects the queue to the top node of the stack.

Table 3 summarizes the results of the rule application and shows the complete list of successors of $POS(QUEUE_0)$. In this way, the search for a node's neighbors along the axes is reduced to one direction, either left or right, or up or down, that will depend on the topological relation that introduced the node in the feature set.

6 Feature Selection Algorithm

At each generation, we compute the Cartesian product of the current feature tuple $Feat_i$ and the set defined by its neighbors. We define the set of candidate tuples $CandFeat_{i+1}$ at generation $i+1$ as:

$$CandFeat_{i+1} = \{Feat_i\} \times SUCC(Feat_i),$$

where we have $Card(CandFeat_{i+1}) = Card(SUCC(Feat_i))$.

The members of $CandFeat_{i+1}$ are ranked according to their parsing score on the development

Data structures				Navigation paths					
$STACK_0$	$STACK_n, n>0$	$QUEUE_0$	$QUEUE_n, n>0$	*h*	*lc, rc*	*ls*	*rs*	*pw*	*fw*
PLD	PLD	PL	PL	h				h	h
PLD h	PLD h				lc	lc	lc	lc	lc
PLD lc	PLD lc	PLD lc			rc	rc	rc	rc	rc
PLD rc	PLD rc	PLD rc		ls	ls	ls		ls	ls
PLD ls	PLD ls			rs	rs		rs	rs	rs
PLD rs	PLD rs			pw	pw	pw	pw	pw	
PLD pw	PLD pw	PLD pw		fw	fw	fw	fw		fw
PLD fw	PLD fw								
PLD $STACK_1$	PLD $STACK_{n+1}$	PL $QUEUE_1$	PL $QUEUE_{n+1}$						
PL $QUEUE_0$		PLD $STACK_0$							

Table 2: Rules to compute the successors of a node. For each node category given in row 2, the procedure adds the features in the column headed by the category. PLD stands for the *POS*, *LEX*, and *DEP* attributes. In the right-hand side of the table, the category corresponds to the last instruction of the navigation path, if it exists, for instance *pw* in the feature $POS(STACK_1\ lc\ pw)$. We read the six successors of this node in the fifth column headed by *pw*: $STACK_1\ lc\ pw\ h$, $STACK_1\ lc\ pw\ lc$, $STACK_1\ lc\ pw\ rc$, $STACK_1\ lc\ pw\ ls$, $STACK_1\ lc\ pw\ rs$, and $STACK_1\ lc\ pw\ pw$. We then apply all the attributes to these destination nodes to generate the features.

Initial feature	POS	QUEUE	0	
Successors	LEX	QUEUE	0	
	PLD	QUEUE	0	lc
	PLD	QUEUE	0	rc
	PLD	QUEUE	0	pw
	PL	QUEUE	1	
	PLD	STACK	0	

Table 3: Features generated by the successor function $SUCC(\{POS(QUEUE_0)\})$. PLD stands for the three attributes *POS*, *LEX*, and *DEP* of the node; PL for *POS* and *LEX*.

set and when applying a greedy best-first search, $Feat_{i+1}$ is assigned with the tuple yielding the highest score:

$$Feat_{i+1} \leftarrow eval\_best(CandFeat_{i+1}).$$

The procedure is repeated with the immediate neighbors of $Feat_{i+1}$ until the improvement of the score is below a certain threshold.

We extended this greedy version of the discovery with a beam search that retains the *N*-best successors from the candidate set. In our experiments, we used beam widths of 4 and 8.

7 Experimental Setup

In a first experiment, we used the Swedish corpus of the CoNLL-X shared task (Buchholz and Marsi, 2006). In a second experiment, we applied the feature discovery procedure to the English corpus from CoNLL 2008 (Surdeanu et al., 2008), a dependency corpus converted from the Penn Treebank and the Brown corpus. In both experiments, we used the LIBSVM package (Chang and Lin, 2001) with a quadratic kernel, $\gamma = 0.2$, $C = 0.4$, and $\varepsilon = 0.1$. These parameters are identical to Nivre et al. (2006b) to enable a comparison of the scores.

We evaluated the feature candidates on a development set using the labeled and unlabeled attachment scores (LAS and UAS) that we computed with the `eval.pl` script from CoNLL-X. As there was no development set for the Swedish corpus, we created one by picking out every 10th sentence from the training set. The training was then carried out on the remaining part of the set.

8 Feature Discovery on a Swedish Corpus

In a first run, the search was optimized for the UAS. In a second one, we optimized the LAS. We also report the results we obtained subsequently on the CoNLL-X test set as an indicator of how

well the training generalized.

8.1 The First and Second Generations

Table 4 shows the feature performance at the first generation sorted by UAS. The first row shows the two initial feature candidates, $\langle POS(QUEUE_0)\rangle$ and $\langle LEX(QUEUE_0)\rangle$. The third row shows the score produced by the initial features alone. The next rows show the unlabeled and labeled attachment scores with feature pairs combining one of the initial features and the one listed in the row. The combination of $POS(QUEUE_0)$ and $POS(STACK_0)$ yielded the best UAS: 74.02. The second feature improves the performance of $POS(QUEUE_0)$ by more than 30 points from 43.49.

For each generation, we applied a beam search. We kept the eight best pairs as starting states for the second generation and we added their neighboring nodes. Table 5 shows the eight best results out of 38 for the pair $\langle POS(QUEUE_0), POS(STACK_0)\rangle$.

Parent state: $\langle POS(QUEUE_0), POS(STACK_0)\rangle$				
Dev set		Test set		
UAS	LAS	UAS	LAS	Successors
79.50	65.34	79.07	65.86	P QUEUE 1
78.73	66.98	76.04	64.51	L STACK 0 fw
77.42	63.08	74.63	61.86	L QUEUE 1
77.06	64.54	75.28	62.90	L QUEUE 0 pw
76.83	66.01	73.61	63.77	L QUEUE 0
76.63	63.62	74.75	63.17	P STACK 0 fw
76.44	64.24	74.09	62.02	L STACK 0
76.39	63.12	73.99	61.16	L QUEUE 0 lc

Table 5: Ranking the successors of $\langle POS(QUEUE_0), POS(STACK_0)\rangle$ on the Swedish corpus. Out of the 38 successors, we show the eight that yielded the best results. P stands for *POS*, L for *LEX*, and D for *DEP*.

8.2 Optimizing the Unlabeled Attachement Score

We iterated the process over a total of 16 generations. Table 6, left-hand side, shows the list of the best scores for each generation. The scores on the development set increased steadily until generation 13, then reached a plateau, and declined around generation 15. The test set closely followed the development set with values about 1% lower. On this set, we reached a peak performance at generation 12, after which the results decreased.

Table 6, right-hand side, shows the features producing the final score in their order of inclusion in the feature set. As we applied a beam search, a feature listed at generation *i* does not necessary correspond to the highest score for this generation, but belongs to the feature tuple producing the best result at generation 16.

8.3 Optimizing the Labeled Attachement Score

We also applied the feature discovery with a search optimized for the labeled attachment score. This time, we reduced the beam width used in the search from 8 to 4 as we noticed that the candidates between ranks 5 and 8 never contributed to the best scoring feature set for any generation.

We observed a score curve similar to that of the UAS-optimized search. The train set followed the development set with increasing values for each generation but 1-2% lower. The optimal value was obtained at generation 15 with 84.21% for the test set. Then, the score for the test set decreased.

9 Feature Discovery on a Corpus of English

The training and development sets of the CoNLL 2008 corpus contain text from the *Wall Street Journal* exclusively. The test set contains text from the Brown corpus as well as from the *Wall Street Journal*. Table 7 shows the results after 16 generations. We used a beam width of 4 and the tests were optimized for the unlabeled attachment score. As for Swedish, we reached the best scores around generation 14-15. The results on the in-domain test set peaked at 90.89 and exceeded the results on the development set. As expected, the results for the out-of-domain corpus were lower, 87.50, however the drop was limited to 3.4.

10 Discussion and Conclusion

The results we attained with feature set sizes as small as 15 are competitive or better than figures

Parent state		$\langle POS(QUEUE_0)\rangle$			$\langle LEX(QUEUE_0)\rangle$
UAS	LAS	Successors	UAS	LAS	Successors
43.49	26.45	None	42.76	23.56	None
74.02	59.67	POS STACK 0	**65.86**	52.18	POS STACK 0
67.77	54.50	LEX STACK 0	**58.59**	45.51	LEX STACK 0
58.37	41.83	POS QUEUE 0 pw	**51.98**	37.70	POS QUEUE 0 pw
55.28	38.49	LEX QUEUE 0 pw	50.44	29.71	POS QUEUE 1
51.53	30.43	POS QUEUE 1	50.38	35.24	LEX QUEUE 0 pw
51.05	32.66	LEX QUEUE 0 lc	49.37	32.27	POS QUEUE 0
49.71	31.54	POS QUEUE 0 lc	48.91	27.77	LEX QUEUE 1
49.49	29.18	LEX QUEUE 1	48.66	29.91	LEX QUEUE 0 lc
49.37	32.27	LEX QUEUE 0	47.25	28.92	LEX QUEUE 0 rc
48.68	29.34	DEP STACK 0	47.09	28.65	POS QUEUE 0 lc
48.47	30.84	LEX QUEUE 0 rc	46.68	27.08	DEP QUEUE 0 lc
46.77	26.86	DEP QUEUE 0 lc	45.69	27.83	POS QUEUE 0 rc
46.40	29.95	POS QUEUE 0 rc	44.77	26.17	DEP STACK 0
42.27	25.21	DEP QUEUE 0 pw	44.43	26.47	DEP QUEUE 0 rc
41.04	26.56	DEP QUEUE 0 rc	41.87	23.04	DEP QUEUE 0 pw

Table 4: Results of the beam search on the Swedish corpus at the first generation with the two initial feature candidates, $\langle POS(QUEUE_0)\rangle$ and $\langle LEX(QUEUE_0)\rangle$, respectively on the left- and right-hand side of the table. The third row shows the score produced by the initial features alone and the next rows, the figures for the candidate pairs combining the initial feature and the successor listed in the row. The eight best combinations shown in bold are selected for the next generation.

Generation	Dev set		Test set		Features
	UAS	LAS	UAS	LAS	
1	43.49	26.45	45.93	30.19	POS QUEUE 0
2	74.02	59.67	71.60	58.37	POS STACK 0
3	79.50	65.34	79.07	65.86	POS QUEUE 1
4	83.58	71.76	82.75	70.98	LEX STACK 0 fw
5	85.96	76.03	84.82	74.75	LEX STACK 0
6	87.23	77.32	86.34	76.52	LEX QUEUE 0 lc
7	88.42	80.00	87.67	78.99	POS STACK 1
8	89.43	81.56	88.09	80.26	LEX QUEUE 1
9	89.84	83.20	88.69	82.33	LEX QUEUE 0
10	90.23	83.89	89.17	83.31	DEP STACK 0 lc
11	90.49	84.31	89.58	**83.85**	POS STACK 0 fw
12	90.73	84.47	**89.66**	83.83	LEX STACK 0 fw ls
13	90.81	84.60	89.52	83.75	LEX STACK 0 fw ls lc
14	90.81	**84.70**	89.32	83.73	POS STACK 1 h
15	**90.85**	84.67	89.13	83.21	LEX STACK 1 rs
16	90.84	84.68	88.65	82.75	POS STACK 0 fw ls rc

Table 6: Best results for each generation on the Swedish corpus, optimized for UAS. Figures in bold designate the best scores. The right-hand side of the table shows the feature sequence producing the best result at generation 16.

Generation	Dev set		Test set WSJ		Test set Brown		Features
	UAS	LAS	UAS	LAS	UAS	LAS	
1	45.25	33.77	45.82	34.49	52.12	40.70	POS QUEUE 0
2	64.42	55.64	64.71	56.44	71.29	62.41	LEX STACK 0
3	78.62	68.77	78.99	70.30	78.67	65.17	POS QUEUE 1
4	81.83	76.67	82.46	77.82	80.57	72.95	LEX STACK 0 fw
5	84.43	79.78	84.89	80.88	84.03	76.99	POS STACK 0
6	85.95	81.60	86.61	82.93	84.55	77.80	DEP QUEUE 0 lc
7	86.95	82.73	87.73	84.09	85.26	78.48	LEX STACK 1
8	88.03	83.62	88.52	84.74	85.66	78.73	LEX QUEUE 1
9	88.61	84.97	89.15	86.20	86.29	79.86	LEX QUEUE 0
10	89.09	85.43	89.47	86.60	86.43	80.02	POS QUEUE 2
11	89.54	85.87	90.25	87.40	87.00	80.75	POS STACK 0 pw
12	89.95	86.21	90.63	87.77	86.87	80.46	POS QUEUE 3
13	90.26	86.56	90.64	87.80	87.35	80.86	POS STACK 1 pw
14	90.54	86.81	90.71	87.88	**87.50**	81.30	POS QUEUE 0 pw
15	90.61	86.94	**90.89**	**88.11**	87.47	**81.33**	LEX STACK 0 lc
16	**90.65**	**87.00**	90.88	88.09	87.42	81.28	POS STACK 0 pw ls

Table 7: Best results for each generation. English corpus. Selection optimized for UAS.

reported by state-of-the-art transition-based systems. We reached a UAS of 89.66 on the CoNLL-X Swedish corpus. On the same corpus, the top scores reported in the shared task were slightly lower: 89.54 and 89.50. Our best LAS was 84.21, and the two best scores in CoNLL-X were 84.58 and 82.55. Our results for the English corpus from CoNLL 2008 were optimized for an unlabeled attachment score and we obtained 90.89 for the in-domain test set and 87.50 for the out-of-domain one. Our best LAS were 88.11 and 81.33. Official results in CoNLL 2008 only reported the labeled attachment scores, respectively 90.13 and 82.81[1].

We believe these results remarkable. We used a single-parser system as opposed to ensemble systems and the results on the Brown corpus show an excellent resilience and robustness on out-of-domain data. The automatic discovery produced results matching or exceeding comparable systems, although no prior knowledge of the language being analyzed was used and no feature set was provided to the parser.

Although, a systematic search requires no intuitive guessing, it still consumes a considerable machine time. Due to the learning algorithm we use, SVM, training a model takes between 1 and 130 hours depending on the size of the corpus. The number of models to train at each generation corresponds to the number of feature candidates times the beam width. The first generation contains about 15 feature candidates per feature set and since features are only added, the number of candidates can grow to 100 at generation 10.

We believe there is a margin for improvement both in the parsing scores and in the time needed to determine the feature sets. Our scores in Swedish were obtained with models trained on 90% of the training set. They could probably be slightly improved if they had been trained on a complete set. In our experiments, we used three attributes: the part of speech, lexical value, and dependency label of the node. These attributes could be extended to lemmas and grammatical features. SVMs yield a high performance, but they are slow to train. Logistic regression with, for instance, the LIBLINEAR package (Fan et al., 2008) would certainly reduce the exploration time.

[1]Results are not exactly comparable as we used the CoNLL-X evaluation script that gives slightly higher figures.

Acknowledgments

The research leading to these results has received funding from the European community's seventh framework program FP7/2007-2013, challenge 2, cognitive systems, interaction, robotics, under grant agreement No 230902—ROSETTA.

References

Attardi, Giuseppe, Felice Dell'Orletta, Maria Simi, Atanas Chanev, and Massimiliano Ciaramita. 2007. Multilingual dependency parsing and domain adaptation using DeSR. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 1112–1118, Prague, Czech Republic, June. Association for Computational Linguistics.

Buchholz, Sabine and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164, New York City, June. Association for Computational Linguistics.

Chang, Chih-Chung and Chih-Jen Lin. 2001. LIBSVM: a library for support vector machines. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Covington, Michael A. 2001. A fundamental algorithm for dependency parsing. In *Proceedings of the 39th Annual ACM Southeast Conference*, Athens, Georgia.

Fan, Rong-En, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Kunze, Jürgen. 1967. Die Behandlung nichtprojektiver Strukturen bei der syntaktischen Analyse und Synthese des englischen und des deutschen. In *MASPEREVOD-67: Internationales Symposium der Mitgliedsländer des RGW*, pages 2–15, Budapest, 10.–13. Oktober.

Nivre, Joakim and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 99–106, Ann Arbor, June.

Nivre, Joakim, Johan Hall, and Jens Nilsson. 2006a. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of the fifth international conference on Language Resources and Evaluation (LREC2006)*, pages 2216–2219, Genoa, May 24-26.

Nivre, Joakim, Johan Hall, Jens Nilsson, Gülsen Eryigit, and Svetoslav Marinov. 2006b. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL)*, pages 221–225, New York, June, 8-9.

Nivre, Joakim. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT 03)*, pages 149–160, Nancy, 23-25 April.

Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL 2008 shared task on joint parsing of syntactic and semantic dependencies. In *CoNLL 2008: Proceedings of the 12th Conference on Computational Natural Language Learning*, pages 159–177, Manchester, August.

Yamada, Hiroyasu and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT 03)*, pages 195–206, Nancy, 23-25 April.

Zhang, Yue and Stephen Clark. 2009. Transition-based parsing of the Chinese treebank using a global discriminative model. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT 09)*, pages 162–171, Paris, October.

Evaluation of Dependency Parsers on Unbounded Dependencies

Joakim Nivre
Uppsala University
joakim.nivre@lingfil.uu.se

Laura Rimell
Univ. of Cambridge
laura.rimell@cl.cam.ac.uk

Ryan McDonald
Google Inc.
ryanmcd@google.com

Carlos Gómez-Rodríguez
Universidade da Coruña
cgomezr@udc.es

Abstract

We evaluate two dependency parsers, MSTParser and MaltParser, with respect to their capacity to recover unbounded dependencies in English, a type of evaluation that has been applied to grammar-based parsers and statistical phrase structure parsers but not to dependency parsers. The evaluation shows that when combined with simple post-processing heuristics, the parsers correctly recall unbounded dependencies roughly 50% of the time, which is only slightly worse than two grammar-based parsers specifically designed to cope with such dependencies.

1 Introduction

Though syntactic parsers for English are reported to have accuracies over 90% on the Wall Street Journal (WSJ) section of the Penn Treebank (PTB) (McDonald et al., 2005; Sagae and Lavie, 2006; Huang, 2008; Carreras et al., 2008), broad-coverage parsing is still far from being a solved problem. In particular, metrics like attachment score for dependency parsers (Buchholz and Marsi, 2006) and Parseval for constituency parsers (Black et al., 1991) suffer from being an average over a highly skewed distribution of different grammatical constructions. As a result, infrequent yet semantically important construction types could be parsed with accuracies far below what one might expect.

This shortcoming of aggregate parsing metrics was highlighted in a recent study by Rimell et al. (2009), introducing a new parser evaluation corpus containing around 700 sentences annotated with unbounded dependencies in seven different grammatical constructions. This corpus was used to evaluate five state-of-the-art parsers for English, focusing on grammar-based and statistical phrase structure parsers. For example, in the sentence *By Monday, they hope to have a sheaf of* **documents** *both sides can* **trust**., parsers should recognize that there is a dependency between **trust** and **documents**, an instance of object extraction out of a (reduced) relative clause. In the evaluation, the recall of state-of-the-art parsers on this kind of dependency varies from a high of 65% to a low of 1%. When averaging over the seven constructions in the corpus, none of the parsers had an accuracy higher than 61%.

In this paper, we extend the evaluation of Rimell et al. (2009) to two dependency parsers, MSTParser (McDonald, 2006) and MaltParser (Nivre et al., 2006a), trained on data from the PTB, converted to Stanford typed dependencies (de Marneffe et al., 2006), and combined with a simple post-processor to extract unbounded dependencies from the basic dependency tree. Extending the evaluation to dependency parsers is of interest because it sheds light on whether highly tuned grammars or computationally expensive parsing formalisms are necessary for extracting complex linguistic phenomena in practice. Unlike the best performing grammar-based parsers studied in Rimell et al. (2009), neither MSTParser nor MaltParser was developed specifically as a parser for English, and neither has any special mechanism for dealing with unbounded dependencies. Dependency parsers are also often asymptotically faster than grammar-based or constituent parsers, e.g., MaltParser parses sentences in linear time.

Our evaluation ultimately shows that the recall of MSTParser and MaltParser on unbounded dependencies is much lower than the average (un)labeled attachment score for each system. Nevertheless, the two dependency parsers are found to perform only slightly worse than the best grammar-based parsers evaluated in Rimell et al.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 833–841,
Beijing, August 2010

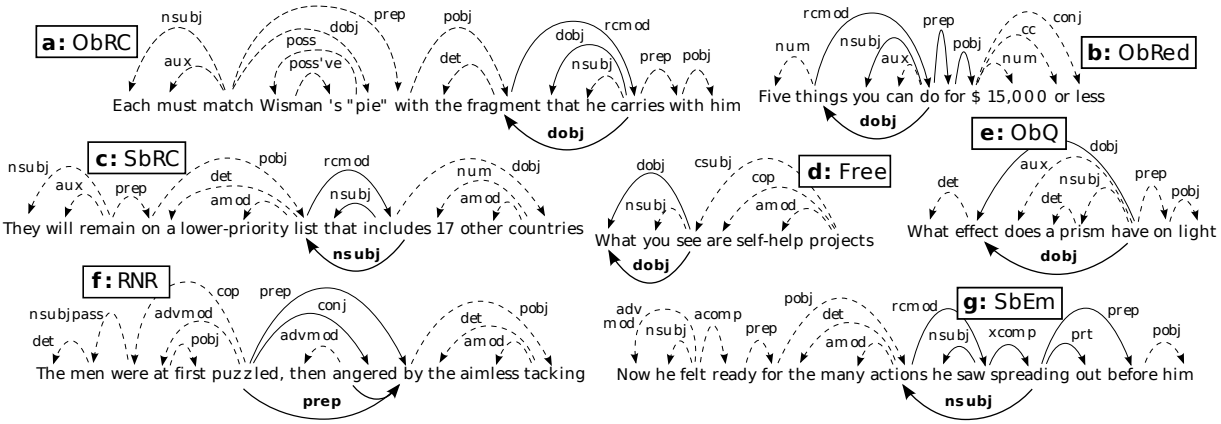


Figure 1: Examples of seven unbounded dependency constructions (**a**–**g**). Arcs drawn *below* each sentence represent the dependencies scored in the evaluation, while the tree *above* each sentence is the Stanford basic dependency representation, with solid arcs indicating crucial dependencies (cf. Section 4). All examples are from the development sets.

(2009) and considerably better than the other statistical parsers in that evaluation. Interestingly, though the two systems have similar accuracies overall, there is a clear distinction between the kinds of errors each system makes, which we argue is consistent with observations by McDonald and Nivre (2007).

2 Unbounded Dependency Evaluation

An unbounded dependency involves a word or phrase interpreted at a distance from its surface position, where an unlimited number of clause boundaries may in principle intervene. The unbounded dependency corpus of Rimell et al. (2009) includes seven grammatical constructions: object extraction from a relative clause (ObRC), object extraction from a reduced relative clause (ObRed), subject extraction from a relative clause (SbRC), free relatives (Free), object questions (ObQ), right node raising (RNR), and subject extraction from an embedded clause (SbEm), all chosen for being relatively frequent and easy to identify in PTB trees. Examples of the constructions can be seen in Figure 1. The evaluation set contains 80 sentences per construction (which may translate into more than 80 dependencies, since sentences containing coordinations may have more than one gold-standard dependency), while the development set contains between 13 and 37 sentences per construction. The data for ObQ sentences was obtained from various years of TREC, and for the rest of the constructions from the WSJ (0-1 and 22-24) and Brown sections of the PTB.

Each sentence is annotated with one or more gold-standard dependency relations representing the relevant unbounded dependency. The gold-standard dependencies are shown as arcs below the sentences in Figure 1. The format of the dependencies in the corpus is loosely based on the Stanford typed dependency scheme, although the evaluation procedure permits alternative representations and does not require that the parser output match the gold-standard exactly, as long as the "spirit" of the construction is correct.

The ability to recover unbounded dependencies is important because they frequently form part of the basic predicate-argument structure of a sentence. Subject and object dependencies in particular are crucial for a number of tasks, including information extraction and question answering. Moreover, Rimell et al. (2009) show that, although individual types of unbounded dependencies may be rare, the unbounded dependency types in the corpus, considered as a class, occur in as many as 10% of sentences in the PTB.

In Rimell et al. (2009), five state-of-the-art parsers were evaluated for their recall on the gold-standard dependencies. Three of the parsers were based on grammars automatically extracted from the PTB: the C&C CCG parser (Clark and Curran, 2007), the Enju HPSG parser (Miyao and Tsujii, 2005), and the Stanford parser (Klein and Manning, 2003). The two remaining systems were the

RASP parser (Briscoe et al., 2006), using a manually constructed grammar and a statistical parse selection component, and the DCU post-processor of PTB parsers (Cahill et al., 2004) using the output of the Charniak and Johnson reranking parser (Charniak and Johnson, 2005). Because of the wide variation in parser output representations, a mostly manual evaluation was performed to ensure that each parser got credit for the constructions it recovered correctly. The parsers were run essentially "out of the box", meaning that the development set was used to confirm input and output formats, but no real tuning was performed. In addition, since a separate question model is available for C&C, this was also evaluated on ObQ sentences. The best overall performers were C&C and Enju, which is unsurprising since they are deep parsers based on grammar formalisms designed to recover just such dependencies. The DCU post-processor performed somewhat worse than expected, often identifying the existence of an unbounded dependency but failing to identify the grammatical class (subject, object, etc.). RASP and Stanford, although not designed to recover such dependencies, nevertheless recovered a subset of them. Performance of the parsers also varied widely across the different constructions.

3 Dependency Parsers

In this paper we repeat the study of Rimell et al. (2009) for two dependency parsers, with the goal of evaluating how parsers based on dependency grammars perform on unbounded dependencies.

MSTParser[1] is a freely available implementation of the parsing models described in McDonald (2006). According to the categorization of parsers in Kübler et al. (2008) it is a *graph-based* parsing system in that core parsing algorithms can be equated to finding directed maximum spanning trees (either projective or non-projective) from a dense graph representation of the sentence. Graph-based parsers typically rely on global training and inference algorithms, where the goal is to learn models in which the weight/probability of correct trees is higher than that of incorrect trees. At inference time a global search is run to find the highest weighted dependency tree. Unfortunately, global inference and learning for graph-based dependency parsing is typically NP-hard (McDonald and Satta, 2007). As a result, graph-based parsers (including MSTParser) often limit the scope of their features to a small number of adjacent arcs (usually two) and/or resort to approximate inference (McDonald and Pereira, 2006).

MaltParser[2] is a freely available implementation of the parsing models described in Nivre et al. (2006a) and Nivre et al. (2006b). MaltParser is categorized as a *transition-based* parsing system, characterized by parsing algorithms that produce dependency trees by transitioning through abstract state machines (Kübler et al., 2008). Transition-based parsers learn models that predict the next state given the current state of the system as well as features over the history of parsing decisions and the input sentence. At inference time, the parser starts in an initial state, then greedily moves to subsequent states – based on the predictions of the model – until a termination state is reached. Transition-based parsing is highly efficient, with run-times often linear in sentence length. Furthermore, transition-based parsers can easily incorporate arbitrary non-local features, since the current parse structure is fixed by the state. However, the greedy nature of these systems can lead to error propagation if early predictions place the parser in incorrect states.

McDonald and Nivre (2007) compared the accuracy of MSTParser and MaltParser along a number of structural and linguistic dimensions. They observed that, though the two parsers exhibit indistinguishable accuracies overall, MSTParser tends to outperform MaltParser on longer dependencies as well as those dependencies closer to the root of the tree (e.g., verb, conjunction and preposition dependencies), whereas MaltParser performs better on short dependencies and those further from the root (e.g., pronouns and noun dependencies). Since long dependencies and those near to the root are typically the last constructed in transition-based parsing systems, it was concluded that MaltParser does suffer from some form of error propagation. On the other hand, the

[1] http://mstparser.sourceforge.net

[2] http://www.maltparser.org

richer feature representations of MaltParser led to improved performance in cases where error propagation has not occurred. However, that study did not investigate unbounded dependencies.

4 Methodology

In this section, we describe the methodological setup for the evaluation, including parser training, post-processing, and evaluation.[3]

4.1 Parser Training

One important difference between MSTParser and MaltParser, on the one hand, and the best performing parsers evaluated in Rimell et al. (2009), on the other, is that the former were never developed specifically as parsers for English. Instead, they are best understood as data-driven parser generators, that is, tools for generating a parser given a training set of sentences annotated with dependency structures. Over the years, both systems have been applied to a wide range of languages (see, e.g., McDonald et al. (2006), McDonald (2006), Nivre et al. (2006b), Hall et al. (2007), Nivre et al. (2007)), but they come with no language-specific enhancements and are not equipped specifically to deal with unbounded dependencies.

Since the dependency representation used in the evaluation corpus is based on the Stanford typed dependency scheme (de Marneffe et al., 2006), we opted for using the WSJ section of the PTB, converted to Stanford dependencies, as our primary source of training data. Thus, both parsers were trained on section 2–21 of the WSJ data, which we converted to Stanford dependencies using the Stanford parser (Klein and Manning, 2003). The Stanford scheme comes in several varieties, but because both parsers require the dependency structure for each sentence to be a tree, we had to use the so-called *basic* variety (de Marneffe et al., 2006).

It is well known that questions are very rare in the WSJ data, and Rimell et al. (2009) found that parsers trained only on WSJ data generally performed badly on the questions included in the evaluation corpus, while the C&C parser equipped with a model trained on a combination of WSJ and question data had much better performance. To investigate whether the performance of MSTParser and MaltParser on questions could also be improved by adding more questions to the training data, we trained one variant of each parser using data that was extended with 3924 questions taken from QuestionBank (QB) (Judge et al., 2006).[4] Since the QB sentences are annotated in PTB style, it was possible to use the same conversion procedure as for the WSJ data. However, it is clear that the conversion did not always produce adequate dependency structures for the questions, an observation that we will return to in the error analysis below.

In comparison to the five parsers evaluated in Rimell et al. (2009), it is worth noting that MSTParser and MaltParser were trained on the same basic data as four of the five, but with a different kind of syntactic representation – dependency trees instead of phrase structure trees or theory-specific representations from CCG and HPSG. It is especially interesting to compare MSTParser and MaltParser to the Stanford parser, which essentially produces the same kind of dependency structures as output but uses the original phrase structure trees from the PTB as input to training.

For our experiments we used MSTParser with the same parsing algorithms and features as reported in McDonald et al. (2006). However, unlike that work we used an atomic maximum entropy model as the second stage arc predictor as opposed to the more time consuming sequence labeler. McDonald et al. (2006) showed that there is negligible accuracy loss when using atomic rather than structured labeling. For MaltParser we used the projective Stack algorithm (Nivre, 2009) with default settings and a slightly enriched feature model. All parsing was projective because the Stanford dependency trees are strictly projective.

[3] To ensure replicability, we provide all experimental settings, post-processing scripts and additional information about the evaluation at http://stp.ling.uu.se/∼nivre/exp/.

[4] QB contains 4000 questions, but we removed all questions that also occurred in the test or development set of Rimell et al. (2009), who sampled their questions from the same TREC QA test sets.

4.2 Post-Processing

All the development and test sets in the corpus of Rimell et al. (2009) were parsed using MSTParser and MaltParser after part-of-speech tagging the input using SVMTool (Giménez and Màrquez, 2004) trained on section 2–21 of the WSJ data in Stanford basic dependency format. The Stanford parser has an internal module that converts the *basic* dependency representation to the *collapsed* representation, which explicitly represents additional dependencies, including unbounded dependencies, that can be inferred from the basic representation (de Marneffe et al., 2006). We performed a similar conversion using our own tool.

Broadly speaking, there are three ways in which unbounded dependencies can be inferred from the Stanford basic dependency trees, which we will refer to as *simple*, *complex*, and *indirect*. In the simple case, the dependency coincides with a single, direct dependency relation in the tree. This is the case, for example, in Figure 1**d**–**e**, where all that is required is that the parser identifies the dependency relation from a governor to an argument (`dobj(see, What)`, `dobj(have, effect)`), which we call the Arg relation; no post-processing is needed.

In the complex case, the dependency is represented by a *path* of direct dependencies in the tree, as exemplified in Figure 1**a**. In this case, it is not enough that the parser correctly identifies the Arg relation `dobj(carries, that)`; it must also find the dependency `rcmod(fragment, carries)`. We call this the Link relation, because it links the argument role inside the relative clause to an element outside the clause. Other examples of the complex case are found in Figure 1**c** and in Figure 1**f**.

In the indirect case, finally, the dependency cannot be defined by a path of labeled dependencies, whether simple or complex, but must be inferred from a larger context of the tree using heuristics. Consider Figure 1**b**, where there is a Link relation (`rcmod(things, do)`), but no corresponding Arg relation inside the relative clause (because there is no overt relative pronoun). However, given the other dependencies, we can infer with high probability that the implicit relation is `dobj`. Another example of the indirect case is in Figure 1**g**. Our post-processing tool performs more heuristic inference for the indirect case than the Stanford parser does (cf. Section 4.3).

In order to handle the complex and indirect cases, our post-processor is triggered by the occurrence of a Link relation (`rcmod` or `conj`) and first tries to add dependencies that are directly implied by a single Arg relation (relations involving relative pronouns for `rcmod`, shared heads and dependents for `conj`). If there is no overt relative pronoun, or the function of the relative pronoun is underspecified, the post-processor relies on the obliqueness hierarchy `subj` $<$ `dobj` $<$ `pobj` and simply picks the first "missing function", unless it finds a clausal complement (indicated by the labels `ccomp` and `xcomp`), in which case it descends to the lower clause and restarts the search there.

4.3 Parser Evaluation

The evaluation was performed using the same criteria as in Rimell et al. (2009). A dependency was considered correctly recovered if the gold-standard head and dependent were correct and the label was an "acceptable match" to the gold-standard label, indicating the grammatical function of the extracted element at least to the level of subject, passive subject, object, or adjunct.

The evaluation in Rimell et al. (2009) took into account a wide variety of parser output formats, some of which differed significantly from the gold-standard. Since MSTParser and MaltParser produced Stanford dependencies for this experiment, evaluation required less manual examination than for some of the other parsers, as was also the case for the output of the Stanford parser in the original evaluation. However, a manual evaluation was still performed in order to resolve questionable cases.

5 Results

The results are shown in Table 1, where the accuracy for each construction is the percentage of gold-standard dependencies recovered correctly. The *Avg* column represents a macroaverage, i.e. the average of the individual scores on the seven constructions, while the *WAvg* column represents

Parser	ObRC	ObRed	SbRC	Free	ObQ	RNR	SbEm	Avg	WAvg
MST	34.1	47.3	78.9	65.5	13.8	**45.4**	**37.6**	46.1	63.4
Malt	**40.7**	**50.5**	**84.2**	**70.2**	16.2	39.7	23.5	46.4	**66.9**
MST-Q					**41.2**			**50.0**	
Malt-Q					31.2			48.5	

Table 1: Parser accuracy on the unbounded dependency corpus.

Parser	ObRC	ObRed	SbRC	Free	ObQ	RNR	SbEm	Avg	WAvg
C&C	**59.3**	62.6	80.0	72.6	**81.2**	**49.4**	22.4	**61.1**	69.9
Enju	47.3	**65.9**	82.1	**76.2**	32.5	47.1	32.9	54.9	**70.9**
MST	34.1	47.3	78.9	65.5	41.2	45.4	**37.6**	50.0	63.4
Malt	40.7	50.5	**84.2**	70.2	31.2	39.7	23.5	48.5	66.9
DCU	23.1	41.8	56.8	46.4	27.5	40.8	5.9	34.6	47.0
RASP	16.5	1.1	53.7	17.9	27.5	34.5	15.3	23.8	34.1
Stanford	22.0	1.1	74.7	64.3	41.2	45.4	10.6	37.0	50.3

Table 2: Parser accuracy on the unbounded dependency corpus. The ObQ score for C&C, MSTParser, and MaltParser is for a model trained with additional questions (without this C&C scored 27.5; MSTParser and MaltParser as in Table 1).

a weighted macroaverage, where the constructions are weighted proportionally to their relative frequency in the PTB. WAvg excludes ObQ sentences, since frequency statistics were not available for this construction in Rimell et al. (2009).

Our first observation is that the accuracies for both systems are considerably below the ∼90% unlabeled and ∼88% labeled attachment scores for English that have been reported previously (McDonald and Pereira, 2006; Hall et al., 2006). Comparing the two parsers, we see that MaltParser is more accurate on dependencies in relative clause constructions (ObRC, ObRed, SbRC, and Free), where argument relations tend to be relatively local, while MSTParser is more accurate on dependencies in RNR and SbEm, which involve more distant relations. Without the additional QB training data, the average scores for the two parsers are indistinguishable, but MSTParser appears to have been better able to take advantage of the question training, since MST-Q performs better than Malt-Q on ObQ sentences. On the weighted average MaltParser scores 3.5 points higher, because the constructions on which it outperforms MSTParser are more frequent in the PTB, and because WAvg excludes ObQ, where MSTParser is more accurate.

Table 2 shows the results for MSTParser and MaltParser in the context of the other parsers evaluated in Rimell et al. (2009).[5] For the parsers which have a model trained on questions, namely C&C, MSTParser, and MaltParser, the figure shown for ObQ sentences is that of the question model. It can be seen that MSTParser and MaltParser perform below C&C and Enju, but above the other parsers, and that MSTParser achieves the highest score on SbEm sentences and MaltParser on SbRC sentences. It should be noted, however, that Table 2 does not represent a direct comparison across all parsers, since most of the other parsers would have benefited from heuristic post-processing of the kind implemented here for MSTParser and MaltParser. This is especially true for RASP, where the grammar explicitly leaves some types of attachment decisions for post-processing. For DCU, improved labeling heuristics would significantly improve performance. It is instructive to compare the dependency parsers to the Stanford parser, which uses the same output representation and has been used to prepare the training data for our experiments. Stanford has very low recall on ObRed and SbEm, the categories where heuristic inference plays the largest role, but mirrors MSTParser for most other categories.

6 Error Analysis

We now proceed to a more detailed error analysis, based on the development sets, and classify

[5] The average scores reported differ slightly from those in Rimell et al. (2009), where a microaverage (i.e., average over all dependencies in the corpus, regardless of construction) was reported.

the errors made by the parsers into three categories: A *global* error is one where the parser completely fails to build the relevant clausal structure – the relative clause in ObRC, ObRed, SbRC, Free, SbEmb; the interrogative clause in ObQ; and the clause headed by the higher conjunct in RNR – often as a result of surrounding parsing errors. When a global error occurs, it is usually meaningless to further classify the error, which means that this category excludes the other two. An Arg error is one where the parser has constructed the relevant clausal structure but fails to find the Arg relation – in the simple and complex cases – or the set of surrounding Arg relations needed to infer an implicit Arg relation – in the indirect case (cf. Section 4.2). A Link error is one where the parser fails to find the crucial Link relation – `rcmod` in ObRC, ObRed, SbRC, SbEmb; `conj` in RNR (cf. Section 4.2). Link errors are not relevant for Free and ObQ, where all the crucial relations are clause-internal.

Table 3 shows the frequency of different error types for MSTParser (first) and MaltParser (second) in the seven development sets. First of all, we can see that the overall error distribution is very similar for the two parsers, which is probably due to the fact that they have been trained on exactly the same data with exactly the same annotation (unlike the five parsers previously evaluated). However, there is a tendency for MSTParser to make fewer Link errors, especially in the relative clause categories ObRC, ObRed and SbRC, which is compatible with the observation from the test results that MSTParser does better on more global dependencies, while MaltParser has an advantage on more local dependencies, although this is not evident from the statistics from the relatively small development set.

Comparing the different grammatical constructions, we see that Link errors dominate for the relative clause categories ObRC, ObRed and SbRC, where the parsers make very few errors with respect to the internal structure of the relative clauses (in fact, no errors at all for MaltParser on SbRC). This is different for SbEm, where the analysis of the argument structure is more complex, both because there are (at least) two clauses involved and because the unbounded dependency

Type	Global	Arg	Link	A+L	Errors	# Deps
ObRC	0/1	1/1	7/11	5/3	13/16	20
ObRed	0/1	0/1	6/7	3/4	9/13	23
SbRC	2/1	1/0	7/13	0/0	10/14	43
Free	2/1	3/5	–	–	5/6	22
ObQ	4/7	13/13	–	–	17/20	25
RNR	6/4	4/6	0/0	4/5	14/15	28
SbEm	3/4	3/2	0/0	3/3	9/9	13

Table 3: Distribution of error types in the development sets; frequencies for MSTParser listed first and MaltParser second. The columns Arg and Link give frequencies for Arg/Link errors occurring without the other error type, while A+L give frequencies for joint Arg and Link errors.

can only be inferred indirectly from the basic dependency representation (cf. Section 4.2). Another category where Arg errors are frequent is RNR, where all such errors consist in attaching the relevant dependent to the second conjunct instead of to the first.[6] Thus, in the example in Figure 1**f**, both parsers found the `conj` relation between **puzzled** and **angered** but attached **by** to the second verb.

Global errors are most frequent for RNR, probably indicating that coordinate structures are difficult to parse in general, and for ObQ (especially for MaltParser), probably indicating that questions are not well represented in the training set even after the addition of QB data.[7] As noted in Section 4.1, this may be partly due to the fact that conversion to Stanford dependencies did not seem to work as well for QB as for the WSJ data. Another problem is that the part-of-speech tagger used was trained on WSJ data only and did not perform as well on the ObQ data. Uses of *What* as a determiner were consistently mistagged as pronouns, which led to errors in parsing. Thus, for the example in Figure 1**e**, both parsers produced the correct analysis except that, because of the tagging error, they treated **What** rather than **effect** as the head of the *wh*-phrase, which counts as an error in the evaluation.

In order to get a closer look specifically at the Arg errors, Table 4 gives the confusion matrix

[6] In the Stanford scheme, an argument or adjunct must be attached to the first conjunct in a coordination to indicate that it belongs to both conjuncts.

[7] Parsers trained without QB had twice as many global errors.

	Sb	Ob	POb	EmSb	EmOb	Other	Total
Sb	–	0/0	0/0	0/0	0/0	2/1	2/1
Ob	2/3	–	0/0	0/1	0/0	4/2	6/6
POb	2/0	7/5	–	0/0	0/0	5/8	14/13
EmSb	1/1	4/2	0/0	–	0/0	1/2	6/5
EmOb	0/0	3/1	0/0	0/0	–	1/6	4/7
Total	5/4	14/8	0/0	0/1	0/0	13/19	32/32

Table 4: Confusion matrix for Arg errors (excluding RNR and using parsers trained on QB for ObQ); frequencies for MSTParser listed first and MaltParser second. The column Other covers errors where the function is left unspecified or the argument is attached to the wrong head.

for such errors, showing which grammatical functions are mistaken for each other, with an extra category Other for cases where the function is left unspecified by the parser or the error is an attachment error rather than a labeling error (and excluding the RNR category because of the special nature of the Arg errors in this category). The results again confirm that the two parsers make very few errors on subjects and objects clause-internally. The few cases where an object is mistaken as a subject occur in ObQ, where both parsers perform rather poorly in general. By contrast, there are many more errors on prepositional objects and on embedded subjects and objects. We believe an important part of the explanation for this pattern is to be found in the Stanford dependency representation, where subjects and objects are marked as such but all other functions realized by *wh* elements are left unspecified (using the generic *rel* dependency), which means that the recovery of these functions currently has to rely on heuristic rules as described in Section 4.2. Finally, we think it is possible to observe the tendency for MaltParser to be more accurate at local labeling decisions – reflected in fewer cross-label confusions – and for MSTParser to perform better on more distant attachment decisions – reflected in fewer errors in the Other category (and in fewer Link errors).

7 Conclusion

In conclusion, the capacity of MSTParser and MaltParser to recover unbounded dependencies is very similar on the macro and weighted macro level, but there is a clear distinction in their strengths – constructions involving more distant dependencies such as ObQ, RNR and SbEm for MSTParser and constructions with more locally defined configurations such as ObRC, ObRed, SbRC and Free for MaltParser. This is a pattern that has been observed in previous evaluations of the parsers and can be explained by the global learning and inference strategy of MSTParser and the richer feature space of MaltParser (McDonald and Nivre, 2007).

Perhaps more interestingly, the accuracies of MSTParser and MaltParser are only slightly below the best performing systems in Rimell et al. (2009) – C&C and Enju. This is true even though MSTParser and MaltParser have not been engineered specifically for English and lack special mechanisms for handling unbounded dependencies, beyond the simple post-processing heuristic used to extract them from the output trees. Thus, it is reasonable to speculate that the addition of such mechanisms could lead to computationally lightweight parsers with the ability to extract unbounded dependencies with high accuracy.

Acknowledgments

We thank Marie-Catherine de Marneffe for great help with the Stanford parser and dependency scheme, Lluís Màrquez and Jesús Giménez for great support with SVMTool, Josef van Genabith for sharing the QuestionBank data, and Stephen Clark and Mark Steedman for helpful comments on the evaluation process and the paper. Laura Rimell was supported by EPSRC grant EP/E035698/1 and Carlos Gómez-Rodríguez by MEC/FEDER (HUM2007-66607-C04) and Xunta de Galicia (PGIDIT07SIN005206PR, Redes Galegas de PL e RI e de Ling. de Corpus, Bolsas Estadas INCITE/FSE cofinanced).

References

Black, E., S. Abney, D. Flickinger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of 4th DARPA Workshop*, 306–311.

Briscoe, T., J. Carroll, and R. Watson. 2006. The second release of the RASP system. In *Proceedings*

of the COLING/ACL 2006 Interactive Presentation Sessions, 77–80.

Buchholz, S. and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of CoNLL*, 149–164.

Cahill, A., M. Burke, R. O'Donovan, J. Van Genabith, and A. Way. 2004. Long-distance dependency resolution in automatically acquired wide-coverage PCFG-based LFG approximations. In *Proceedings of ACL*, 320–327.

Carreras, X., M. Collins, and T. Koo. 2008. TAG, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *Proceedings of CoNLL*, 9–16.

Charniak, E. and M. Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of ACL*, 173–180.

Clark, S. and J. R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33:493–552.

de Marneffe, M.-C., B. MacCartney, and C. D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*.

Giménez, J. and L. Màrquez. 2004. SVMTool: A general POS tagger generator based on support vector machines. In *Proceedings of LREC*.

Hall, J., J. Nivre, and J. Nilsson. 2006. Discriminative classifiers for deterministic dependency parsing. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, 316–323.

Hall, J., J. Nilsson, J. Nivre, G. Eryiğit, B. Megyesi, M. Nilsson, and M. Saers. 2007. Single malt or blended? A study in multilingual parser optimization. In *Proceedings of the CoNLL Shared Task*.

Huang, L. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL*, 586–594.

Judge, J., A. Cahill, and J. van Genabith. 2006. QuestionBank: Creating a corpus of parse-annotated questions. In *Proceedings of COLING-ACL*, 497–504.

Klein, D. and C. D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL*, 423–430.

Kübler, S., R. McDonald, and J. Nivre. 2008. *Dependency Parsing*. Morgan and Claypool.

McDonald, R. and J. Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of EMNLP-CoNLL*, 122–131.

McDonald, R. and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*, 81–88.

McDonald, R. and G. Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proceedings of IWPT*, 122–131.

McDonald, R., K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL*, 91–98.

McDonald, R., K. Lerman, and F. Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of CoNLL*, 216–220.

McDonald, R.. 2006. *Discriminative Learning and Spanning Tree Algorithms for Dependency Parsing*. Ph.D. thesis, University of Pennsylvania.

Miyao, Y. and J. Tsujii. 2005. Probabilistic disambiguation models for wide-coverage HPSG parsing. In *Proceedings of ACL*, 83–90.

Nivre, J., J. Hall, and J. Nilsson. 2006a. MaltParser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC*, 2216–2219.

Nivre, J., J. Hall, J. Nilsson, G. Eryiğit, and S. Marinov. 2006b. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of CoNLL*, 221–225.

Nivre, J., J. Hall, J. Nilsson, A. Chanev, G. Eryiğit, S. Kübler, S. Marinov, and E. Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13:95–135.

Nivre, J. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of ACL-IJCNLP*, 351–359.

Rimell, L., S. Clark, and M. Steedman. 2009. Unbounded dependency recovery for parser evaluation. In *Proceedings EMNLP*, 813–821.

Sagae, K. and A. Lavie. 2006. Parser combination by reparsing. In *Proceedings of NAACL HLT: Short Papers*, 129–132.

Co-STAR: A Co-training Style Algorithm for Hyponymy Relation Acquisition from Structured and Unstructured Text

Jong-Hoon Oh, Ichiro Yamada, Kentaro Torisawa, and Stijn De Saeger
Language Infrastructure Group, MASTAR Project,
National Institute of Information and Communications Technology (NICT)
{rovellia,iyamada,torisawa,stijn}@nict.go.jp

Abstract

This paper proposes a co-training style algorithm called Co-STAR that acquires hyponymy relations simultaneously from structured and unstructured text. In Co-STAR, two independent processes for hyponymy relation acquisition – one handling structured text and the other handling unstructured text – collaborate by repeatedly exchanging the knowledge they acquired about hyponymy relations. Unlike conventional co-training, the two processes in Co-STAR are applied to different source texts and training data. We show the effectiveness of this algorithm through experiments on large-scale hyponymy-relation acquisition from Japanese Wikipedia and Web texts. We also show that Co-STAR is robust against noisy training data.

1 Introduction

Acquiring semantic knowledge, especially semantic relations between lexical terms, is regarded as a crucial step in developing high-level natural language applications. This paper proposes Co-STAR (a **Co**-training **ST**yle **A**lgorithm for hyponymy **R**elation acquisition from structured and unstructured text). Similar to co-training (Blum and Mitchell, 1998), two hyponymy relation extractors in Co-STAR, one for structured and the other for unstructured text, iteratively collaborate to boost each other's performance.

Many algorithms have been developed to automatically acquire semantic relations from structured and unstructured text. Because term pairs are encoded in structured and unstructured text in different styles, different kinds of evidence have been used for semantic relation acquisition:

Evidence from unstructured text: lexico-syntactic patterns and distributional similarity (Ando et al., 2004; Hearst, 1992; Pantel et al., 2009; Snow et al., 2006; De Saeger et al., 2009; Van Durme and Pasca, 2008);

Evidence from structured text: topic hierarchy, layout structure of documents, and HTML tags (Oh et al., 2009; Ravi and Pasca, 2008; Sumida and Torisawa, 2008; Shinzato and Torisawa, 2004).

Recently, researchers have used both structured and unstructured text for semantic-relation acquisition, with the aim of exploiting such different kinds of evidence at the same time. They either tried to improve semantic relation acquisition by putting the different evidence together into a single classifier (Pennacchiotti and Pantel, 2009) or to improve the coverage of semantic relations by combining and ranking the semantic relations obtained from two source texts (Talukdar et al., 2008).

In this paper we propose an algorithm called Co-STAR. The main contributions of this work can be summarized as follows.

- Co-STAR is a semi-supervised learning method composed of two parallel and iterative processes over structured and unstructured text. It was inspired by bilingual co-training, which is a framework for hyponymy relation acquisition from source texts in two languages (Oh et al., 2009). Like bilingual co-training, two processes in Co-STAR operate independently on structured text and unstructured text. These two processes are trained in a supervised manner with their initial training data and then each of them tries to enlarge the existing training data of the other by iteratively exchanging what they

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 842–850,
Beijing, August 2010

have learned (more precisely, by transferring reliable classification results on common instances to one another) (see Section 4 for comparison Co-STAR and bilingual co-training). Unlike the ensemble semantic framework (Pennacchiotti and Pantel, 2009), Co-STAR does not have a single "*master*" classifier or ranker to integrate the different evidence found in structured and unstructured text. We experimentally show that, at least in our setting, Co-STAR works better than a single "*master*" classifier.

- Common relation instances found in both structured and unstructured text act as a *communication channel* between the two acquisition processes. Each process in Co-STAR classifies common relation instances and then transfers its high-confidence classification results to training data of the other process (as shown in Fig. 1), in order to improve classification results of the other process. Moreover, the efficiency of this exchange can be boosted by increasing the "bandwidth" of this channel. For this purpose each separate acquisition process automatically generates a set of relation instances that are likely to be *negative*. In our experiments, we show that the above idea proved highly effective.

- Finally, the acquisition algorithm we propose is robust against noisy training data. We show this by training one classifier in Co-STAR with manually labeled data and training the other with automatically generated but noisy training data. We found that Co-STAR performs well in this setting. This issue is discussed in Section 6.

This paper is organized as follows. Sections 2 and 3 precisely describe our algorithm. Section 4 describes related work. Sections 5 and 6 describe our experiments and present their results. Conclusions are drawn in Section 7.

2 Co-STAR

Co-STAR consists of two processes that simultaneously but independently extract and classify

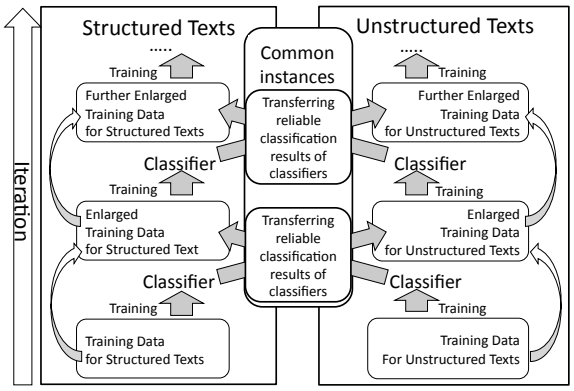


Figure 1: Concept of *Co-STAR*.

hyponymy relation instances from structured and unstructured text. The core of Co-STAR is the collaboration between the two processes, which continually exchange and compare their acquired knowledge on hyponymy relations. This collaboration is made possible through common instances shared by both processes. These common instances are classified separately by each process, but high-confidence classification results by one process can be transferred as new training data to the other.

2.1 Common Instances

Let S and U represent a source (i.e. corpus) of structured and unstructured text, respectively. In this paper, we use the hierarchical layout of Wikipedia articles and the Wikipedia category system as structured text S (see Section 3.1), and a corpus of ordinary Web pages as unstructured text U. Let X_S and X_U denote a set of hyponymy relation candidates extracted from S and U, respectively. X_S is extracted from the hierarchical layout of Wikipedia articles (Oh et al., 2009) and X_U is extracted by lexico-syntactic patterns for hyponymy relations (i.e., *hyponym* such as *hyponymy*) (Ando et al., 2004) (see Section 3 for a detailed explanation)

We define two types of common instances, called "*genuine*" common instances (G) and "*virtual*" common instances (V). The set of common instances is denoted by $Y = G \cup V$. Genuine common instances are hyponymy relation candidates found in both S and U ($G = X_S \cap X_U$). On

the other hand, term pairs are obtained as virtual common instances when:

- 1) they are extracted as hyponymy relation candidates in either S or U and;
- 2) they do not seem to be a hyponymy relation in the other text

The first condition corresponds to $X_S \oplus X_U$. Term pairs satisfying the second condition are defined as R_S and R_U, where $R_S \cap X_S = \phi$ and $R_U \cap X_U = \phi$.

R_S contains term pairs that are found in the Wikipedia category system but neither term appears as ancestor of the other[1]. For example, (*nutrition,protein*) and (*viruses,viral disease*), respectively, hold a category-article relation, where *nutrition* is not ancestor of *viruses* and vice versa in the Wikipedia category system. Here, term pairs, such as (*nutrition, viruses*) and (*viral disease, nutrition*), can be ones in R_S.

R_U is a set of term pairs extracted from U when:

- they are not hyponymy relation candidates in X_U and;
- they regularly co-occur in the same sentence as arguments of the same verb (e.g., A *cause* B or A *is made by* B);

As a result, term pairs in R_U are thought as holding some other semantic relations (e.g., A and B in "A *cause* B" may hold a cause/effect relation) than hyponymy relation. Finally, virtual common instances are defined as:

- $V = (X_S \oplus X_U) \cap (R_S \cup R_U)$

The virtual common instances, from the viewpoint of either S or U, are unlikely to hold a hyponymy relation even if they are extracted as hyponymy relation candidates in the other text. Thus many virtual common instances would be a negative example for hyponymy relation acquisition. On the other hand, genuine common instances (hyponymy relation candidates found in both S and U) are more likely to hold a hyponymy relation than virtual common instances.

In summary, genuine and virtual common instances can be used as different ground for collaboration as well as broader collaboration channel between the two processes than genuine common instances used alone.

[1] A term pair often holds a hyponymy relation if one term in the term pair is a parent of the other in the Wikipedia category system (Suchanek et al., 2007).

2.2 Algorithm

We assume that classifier c assigns class label $cl \in \{yes, no\}$ ("yes" (hyponymy relation) or "no" (not a hyponymy relation)) to instances in $x \in X$ with confidence value $r \in \mathbb{R}^+$, a non-negative real number. We denote the classification result by classifier c as $c(x) = (x, cl, r)$. We used support vector machines (SVMs) in our experiments and the absolute value of the distance between a sample and the hyperplane determined by the SVMs as confidence value r.

Input: Common instances ($Y = G \cup V$) and the initial training data (L_S^0 and L_U^0)
Output: Two classifiers (c_S^n and c_U^n)
$i = 0$
repeat
  $c_S^i := LEARN(L_S^i)$
  $c_U^i := LEARN(L_U^i)$
  $CR_S^i := \{c_S^i(y) | y \in Y, y \notin L_S^i \cup L_U^i\}$
  $CR_U^i := \{c_U^i(y) | y \in Y, y \notin L_S^i \cup L_U^i\}$
  for each $(y, cl_S, r_S) \in TopN(CR_S^i)$ and $(y, cl_U, r_U) \in CR_U^i$ **do**
    if ($r_S > \alpha$ and $r_U < \beta$) or ($r_S > \alpha$ and $cl_S = cl_U$) **then**
      $L_U^{(i+1)} := L_U^{(i+1)} \cup \{(y, cl_S)\}$
    end if
  end for
  for each $(y, cl_U, r_U) \in TopN(CR_U^i)$ and $(y, cl_S, r_S) \in CR_S^i$ **do**
    if ($r_U > \alpha$ and $r_S < \beta$) or ($r_U > \alpha$ and $cl_S = cl_U$) **then**
      $L_S^{(i+1)} := L_S^{(i+1)} \cup \{(y, cl_U)\}$
    end if
  end for
  $i = i + 1$
until stop condition is met

Figure 2: Co-STAR algorithm

The Co-STAR algorithm is given in Fig. 2. The algorithm is interpreted as an iterative procedure 1) to train classifiers (c_U^i, c_S^i) with the existing training data (L_S^i and L_U^i) and 2) to select new training instances from the common instances to be added to existing training data. These are repeated until stop condition is met.

In the initial stage, two classifiers c_S^0 and c_U^0 are trained with manually prepared labeled instances (or training data) L_S^0 and L_U^0, respectively. The learning procedure is denoted by $c = LEARN(L)$ in lines 5–6, where c is a resulting classifier. Then c_S^i and c_U^i are applied to classify common instances in Y (lines 7–8). We denote CR_S^i as a set of the classification results of c_S^i for common instances, which are not included in the current training data $L_S^i \cup L_U^i$. Lines 9–13 describe a way of selecting instances in CR_S^i to be added to the existing training data in U. During the selection, c_S^i acts as a teacher and c_U^i as a student. $TopN(CR_S^i)$ is a set of $c_S^i(y) = (y, cl_S, r_S)$, whose r_S is the top-N highest in CR_S^i. (In our experiments, $N = 900$.) The teacher instructs his student the class label of y if the teacher can decide the class label of y with a certain level of confidence ($r_S > \alpha$) and the student satisfies one of the following two conditions:

- the student agrees with the teacher on class label of y ($cl_S = cl_U$) or
- the student's confidence in classifying y is low ($r_U < \beta$)

$r_U < \beta$ enables the teacher to instruct his student in spite of their disagreement over a class label. If one of the two conditions is satisfied, (y, cl_S) is added to existing labeled instances $L_U^{(i+1)}$. The roles are reversed in lines 14–18, so that c_U^i becomes the teacher and c_S^i the student.

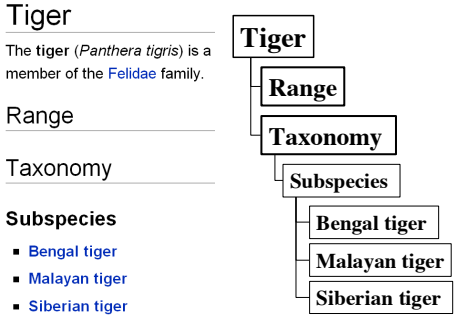
The iteration stops if the change in the difference between the two classifiers is stable enough. The stability is estimated by $d(c_S^i, c_U^i)$ in Eq. (1), where σ^i represents the change in the average difference between the confidence values of the two classifiers in classifying common instances. We terminate the iteration if $d(c_S^i, c_U^i)$ is smaller than 0.001 in three consecutive rounds (Wang and Zhou, 2007).

$$d(c_S^i, c_U^i) = |\sigma^i - \sigma^{(i-1)}|/|\sigma^{(i-1)}| \quad (1)$$

3 Hyponymy Relation Acquisition

In this section we explain how each process extracts hyponymy relations from its respective text source either Wikipedia or Web pages. Each process extracts hyponymy relation candidates (denoted by (*hyper,hypo*) in this section). Because there are many non-hyponymy relations in these candidates[2], we classify hyponymy relation candidates into correct hyponymy relation or not. We used SVMs (Vapnik, 1995) for the classification in this paper.

3.1 Acquisition from Wikipedia



(a) Layout structure (b) Tree structure

Figure 3: Example borrowed from Oh et al. (2009): Layout and tree structures of Wikipedia article TIGER

We follow the method in Oh et al. (2009) for acquiring hyponymy relations from the Japanese Wikipedia. Every article is transformed into a tree structure as shown in Fig. 3, based on the items in its hierarchical layout including *title*, *(sub)section headings*, and *list items*. Candidate relations are extracted from this tree structure by regarding a node as a hypernym candidate and all of its subordinate nodes as potential hyponyms of the hypernym candidate (e.g., (TIGER, TAXONOMY) and (TIGER, SIBERIAN TIGER) from Fig. 3). We obtained 1.9×10^7 Japanese hyponymy relation candidates from Wikipedia.

[2]Only 25–30% of candidates was true hyponymy relation in our experiments.

	Type	Description
Feature from Wikipedia ("WikiFeature")	Lexical	Morphemes and POS of *hyper* and *hypo*; *hyper* and *hypo* themselves
	Structure	Distance between *hyper* and *hypo* in a tree structure; Lexical patterns for article or section names, where listed items often appear; Frequently used section headings in Wikipedia (e.g., "Reference"); Layout item type (e.g., section or list); Tree node type (e.g., root or leaf); Parent and children nodes of *hyper* and *hypo*
	Infobox	Attribute type and its value obtained from Wikipedia infoboxes
Feature from Web texts ("WebFeature")	Lexical	Morphemes and POS of *hyper* and *hypo*; *hyper* and *hypo* themselves
	Pattern	Lexico-syntactic patterns applied to *hyper* and *hypo*; PMI score between pattern and hyponymy relation candidate *(hyper,hypo)*
	Collocation	PMI score between *hyper* and *hypo*
	Noun Class	Noun classes relevant to *hyper* and *hypo*

Table 1: Feature sets (WikiFeature and WebFeature): *hyper* and *hypo* represent hypernym and hyponym parts of hyponymy relation candidates, respectively.

As features for classification we used lexical, structure, and infobox information from Wikipedia (WikiFeature), as shown in Table 1. Because they are the same feature sets as those used in Oh et al. (2009), here we just give a brief overview of the feature sets. Lexical features[3] are used to recognize the lexical evidence for hyponymy relations encoded in *hyper* and *hypo*. For example, the common head morpheme *tiger* in (TIGER, BENGAL TIGER) can be used as the lexical evidence. Such information is provided along with the words/morphemes and the parts of speech of *hyper* and *hypo*, which can be multi-word/morpheme nouns.

Structure features provide evidence found in layout or tree structures for hyponymy relations. For example, hyponymy relations (TIGER, BENGAL TIGER) and (TIGER,MALAYAN TIGER) can be obtained from tree structure "(root node, children nodes of *Subspecies*)" in Fig 3.

3.2 Acquisition from Web Texts

As the target for hyponymy relation acquisition from the Web, we used 5×10^7 pages from the TSUBAKI corpus (Shinzato et al., 2008), a 10^8 page Japanese Web corpus that was dependency parsed with KNP (Kurohashi-Nagao Parser) (Kurohashi and Kawahara, 2005). Hyponymy relation candidates are extracted from the corpus based on the lexico-syntactic patterns such as "*hypo* nado *hyper* (*hyper* such as *hypo*)" and "*hypo* to iu *hyper* (*hyper* called *hypo*)" (Ando et al., 2004). We extracted 6×10^6 Japanese hyponymy relation candidates from the Japanese Web texts. Features (WebFeature) used for classification are summarized in Table 1. Similar to the hyponymy relation acquisition from Wikipedia, lexical features are used to recognize the lexical evidence for hyponymy relations.

Lexico-syntactic patterns for hyponymy relation show different coverage and accuracy in hyponymy relation acquisition (Ando et al., 2004). Further if multiple lexico-syntactic patterns support acquisition of hyponymy relation candidates, these candidates are more likely to be actual hyponymy relations. The pattern feature of hyponymy relation candidates is used for these evidence.

We use PMI (point-wise mutual information) of hyponymy relation candidate (*hyper*, *hypo*) as a collocation feature (Pantel and Ravichandran, 2004), where we assume that *hyper* and *hypo* in candidates would frequently co-occur in the same sentence if they hold a hyponymy relation.

Semantic noun classes have been regarded as useful information in semantic relation acquisition (De Saeger et al., 2009). EM-based clustering (Kazama and Torisawa, 2008) is used for obtaining 500 semantic noun classes[4] from 5×10^5 nouns (including single-word and multi-word ones) and their 4×10^8 dependency relations with 5×10^5 verbs and other nouns in our target Web

[3]MeCab (`http://mecab.sourceforge.net/`) was used to provide the lexical features.

[4]Because EM clustering provides a probability distribution over noun class nc, we obtain discrete classes of each noun n with a probability threshold $p(nc|n) \geq 0.2$ (De Saeger et al., 2009).

	Co-training (Blum and Mitchell, 1998)	Bilingual co-training (Oh et al., 2009)	Co-STAR (Proposed method)
Instance space	Same	Different	Almost different
Feature space	Split by human decision	Split by languages	Split by source texts
Common instances	Genuine-common (or All unlabeled) instances	Genuine-common instances (Translatable)	Genuine-common and virtual-common instances

Table 2: Differences among co-training, bilingual co-training, and Co-STAR

corpus. For example, noun class C_{311} includes biological or chemical substances such as *tatou* (*polysaccharide*) and *yuukikagoubutsu* (*organic compounds*). Noun classes (i.e., C_{311}) relevant to *hyper* and *hypo*, respectively, are used as a noun class feature.

4 Related Work

There are two frameworks, which are most relevant to our work – bilingual co-training and ensemble semantics.

The main difference between bilingual co-training and Co-STAR lies in an instance space. In bilingual co-training, instances are in different spaces divided by languages while, in Co-STAR, many instances are in different spaces divided by their source texts. Table 2 shows differences between co-training, bilingual co-training and Co-STAR.

Ensemble semantics is a relation acquisition framework, where semantic relation candidates are extracted from multiple sources and a single ranker ranks or classifies the candidates in the final step (Pennacchiotti and Pantel, 2009). In ensemble semantics, one ranker is in charge of ranking all candidates extracted from multiple sources; while one classifier classifies candidates extracted from one source in Co-STAR.

5 Experiments

We used the July version of Japanese Wikipedia (jawiki-20090701) as structured text. We randomly selected 24,000 hyponymy relation candidates from those identified in Wikipedia and manually checked them. 20,000 of these samples were used as training data for our initial classifier, the rest was equally divided into development and test data for Wikipedia. They are called "WikiSet." As unstructured text, we used 5×10^7 Japanese Web pages in the TSUBAKI corpus (Shinzato et al., 2008). Here, we manually checked 9,500 hyponymy relation candidates selected randomly from Web texts. 7,500 of these were used as training data. The rest was split into development and test data. We named this data "WebSet".

In both classifiers, the development data was used to select the optimal parameters, and the test data was used to evaluate our system. We used TinySVM (TinySVM, 2002) with a polynomial kernel of degree 2 as a classifier. α (the threshold value indicating high confidence), β (the threshold value indicating low confidence), and $TopN$ (the maximum number of training instances to be added to the existing training data in each iteration) were selected through experiments on the development set. The combination of $\alpha = 1$, $\beta = 0.3$, and $TopN$=900 showed the best performance and was used in the following experiments. Evaluation was done by precision (P), recall (R), and F-measure (F).

5.1 Results

We compare six systems. Three of these, B1–B3, show the effect of different feature sets ("WikiFeature" and "WebFeature" in Table 1) and different training data. We trained two separate classifiers in B1 and B2, while we integrated feature sets and training data for training a single classifier in B3. The classifiers in these three systems are trained with manually prepared training data ("WikiSet" and "WebSet"). For the purpose of our experiment, we consider **B3** as the closest possible approximation of the ensemble semantics framework (Pennacchiotti and Pantel, 2009).

- **B1** consists of two completely independent classifiers. Both S and U classifiers are trained and tested on their own feature and data sets (respectively "WikiSet + WikiFeature" and "WebSet + WebFeature").

- **B2** is the same as B1, except that both classifiers are trained with all available training data — WikiSet and WebSet are combined (27,500 training instances in total). However, each classifier only uses its own feature set (WikiFeature or WebFeature)[5].

- **B3** adds a *master* classifier to **B1**. This third classifier is trained on the complete 27,500 training instances (same as **B2**) using all available features from Table 1, including each instance's SVM scores obtained from the two **B1** classifiers[6]. The verdict of the master classifier is considered to be the final classification result.

The other three systems, BICO, Co-B, and Co-STAR (our proposed method), are for comparison between bilingual co-training (Oh et al., 2009) (BICO) and variants of Co-STAR (Co-B and Co-STAR). Especially, we prepared Co-B and Co-STAR to show the effect of different configurations of common instances on the Co-STAR algorithm. We use both B1 and B2 as the initial classifiers of Co-B and Co-STAR. We notate Co-B and Co-STAR without '$*$' when B1 is used as their initial classifier and those with '$*$' when B2 is used.

- **BICO** implements the bilingual co-training algorithm of (Oh et al., 2009), in which two processes collaboratively acquire hyponymy relations in two *different languages*. For BICO, we prepared 20,000 English and 20,000 Japanese training samples (Japanese ones are the same as training data in the WikiSet) by hand.

- **Co-B** is a variant of Co-STAR that uses only the genuine-common instances as common instances (67,000 instances)[7], to demonstrate the effectiveness of the virtual common instances.

- **Co-STAR** is our proposed method, which uses both genuine-common and virtual-common instances (643,000 instances in total).

	WebSet			WikiSet		
	P	R	F	P	R	F
B1	84.3	65.2	73.5	87.8	74.7	80.7
B2	83.4	69.6	75.9	87.4	79.5	83.2
B3	82.2	72.0	76.8	86.1	77.7	81.7
BICO	N/A	N/A	N/A	84.5	**81.8**	83.1
Co-B	**86.2**	63.5	73.2	**89.7**	74.1	81.2
Co-B$*$	85.5	69.9	77.0	89.6	76.5	82.5
Co-STAR	85.9	76.0	80.6	88.0	**81.8**	**84.8**
Co-STAR$*$	83.3	**80.7**	**82.0**	87.6	**81.8**	84.6

Table 3: Comparison of different systems

Table 3 summarizes the result. Features for common instances in Co-B and Co-STAR are prepared in the same way as training data in B2, so that both classifiers can classify the common instances with their trained feature sets.

Comparison between B1–B3 shows that B2 and B3 outperform B1 in F-measure. More training data used in B2–B3 (27,500 instances for both WebSet and WikiSet) results in higher performance than that of B1 (7,500 and 20,000 instances used separately). We think that the lexical features, assigned regardless of source text to instances in B2–B3, are mainly responsible for the performance gain over B1, as they are the least domain-dependent type of features. B2–B3 are composed of different number of classifiers, each of which is trained with different feature sets and training instances. Despite this difference, B2 and B3 showed similar performance in F-measure.

Co-STAR outperformed the algorithm similar to the ensemble semantics framework (B3), although we admit that a more extensive comparison is desirable. Further Co-STAR outperformed BICO. While the manual cost for building the initial training data used in Co-STAR and BICO is hard to quantify, Co-STAR achieves better performance with fewer training data in total (27,500 instances) than BICO (40,000 instances). The difference in performance between Co-B and Co-STAR shows the effectiveness of

[5]Note that training instances from WebSet (or WikiSet) can have WikiFeature (or WebFeature) if they also appear in Wikipedia (or Web corpus). But they can always have lexical feature, the common feature set between WikiFeature and WebFeature.

[6]SVM scores are assigned to the instances in training data in a 10-fold cross validation manner.

[7]Co-B can be considered as conventional co-training (Blum and Mitchell, 1998) in the sense that two classifiers collaborate through actual common instances.

the automatically generated virtual-common instances. From these comparison, we can see that virtual-common instances coupled with genuine-common instances can be leveraged to enable more effective collaboration between the two classifiers in Co-STAR.

As a result, our proposed method outperforms the others in F-measure by 1.4–8.5%. We obtained 4.3×10^5 hyponymy relations from Web texts and 4.6×10^6 ones from Wikipedia[8].

6 Co-STAR with Automatically Generated Training Data

For Co-STAR, we need two sets of manually prepared training data, one for structured text and the other for unstructured text. As in any other supervised system, the cost of preparing the training data is an important issue. We therefore investigated whether Co-STAR can be trained for a lower cost by generating more of its training data automatically.

We automatically built training data for Web texts by using definition sentences[9] and category names in the Wikipedia articles, while we stuck to manually prepared training data for Wikipedia. To obtain hypernyms from Wikipedia article names, we used definition-specific lexico-syntactic patterns such as "*hyponym* is *hypernym*" and "*hyponym* is a type of *hypernym*" (Kazama and Torisawa, 2007; Sumida and Torisawa, 2008). Then, we extracted hyponymy relations consisting of pairs of Wikipedia category names and their member articles when the Wikipedia category name and the hypernym obtained from the definition of the Wikipedia article shared the same head word. Next, we selected a subset of the extracted hyponymy relations that are also hyponymy relation candidates in Web texts, as positive instances for hyponymy relation acquisition from Web text. We obtained around 15,000 positive instances in this way. Negative instances were chosen from virtual-common instances, which also originated from the Wikipedia category system and hyponymy relation candidates in Web texts (around 293,000 instances).

The automatically built training data was noisy and its size was much bigger than manually prepared training data in WebSet. Thus 7,500 instances as training data (the same number of manually built training data in WebSet) were randomly chosen from the positive and negative instances with a positive:negative ratio of 1:4[10].

	WebSet			WikiSet		
	P	R	F	P	R	F
B1	81.0	47.6	60.0	**87.8**	74.7	80.7
B2	80.0	55.4	65.5	87.1	79.5	83.1
B3	82.0	33.7	47.8	87.1	75.6	81.0
Co-STAR	**82.2**	60.8	69.9	87.3	80.7	83.8
Co-STAR$*$	79.2	**69.6**	**74.1**	87.0	**81.8**	**84.4**

Table 4: Results with automatically generated training data

With the automatically built training data for Web texts and manually prepared training data for Wikipedia, we evaluated B1–B3 and Co-STAR, which are the same systems in Table 3. The results in Table 4 are encouraging. Co-STAR was robust even when faced with noisy training data. Further Co-STAR showed better performance than B1–B3, although its performance in Table 4 dropped a bit compared to Table 3. This result shows that we can reduce the cost of manually preparing training data for Co-STAR with only small loss of the performance.

7 Conclusion

This paper proposed Co-STAR, an algorithm for hyponymy relation acquisition from structured and unstructured text. In Co-STAR, two independent processes of hyponymy relation acquisition from structured texts and unstructured texts, collaborate in an iterative manner through common instances. To improve this collaboration, we introduced virtual-common instances.

Through a series of experiments, we showed that Co-STAR outperforms baseline systems and virtual-common instances can be leveraged to achieve better performance. We also showed that Co-STAR is robust against noisy training data, which requires less human effort to prepare it.

[8] We obtained them with 90% precision by setting the SVM score threshold to 0.23 for Web texts and 0.1 for Wikipedia.

[9] The first sentences of Wikipedia articles.

[10] We select the ratio by testing different ratio from 1:2 to 1:5 with our development data in WebSet and B1.

References

Ando, Maya, Satoshi Sekine, and Shun Ishiza. 2004. Automatic extraction of hyponyms from Japanese newspaper using lexico-syntactic patterns. In *Proc. of LREC '04*.

Blum, Avrim and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT' 98: Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100.

De Saeger, Stijn, Kentaro Torisawa, Jun'ichi Kazama, Kow Kuroda, and Masaki Murata. 2009. Large scale relation acquisition using class dependent patterns. In *Proc. of ICDM 2009*, pages 764–769.

Hearst, Marti A. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics*, pages 539–545.

Kazama, Jun'ichi and Kentaro Torisawa. 2007. Exploiting Wikipedia as external knowledge for named entity recognition. In *Proc. of Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 698–707.

Kazama, Jun'ichi and Kentaro Torisawa. 2008. Inducing gazetteers for named entity recognition by large-scale clustering of dependency relations. In *Proceedings of ACL-08: HLT*, pages 407–415.

Kurohashi, Sadao and Daisuke Kawahara. 2005. KNP (Kurohashi-Nagao Parser) 2.0 users manual.

Oh, Jong-Hoon, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009. Bilingual co-training for monolingual hyponymy-relation acquisition. In *Proc. of ACL-09: IJCNLP*, pages 432–440.

Pantel, Patrick and Deepak Ravichandran. 2004. Automatically labeling semantic classes. In *Proc. of HLT-NAACL '04*, pages 321–328.

Pantel, Patrick, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas. 2009. Web-scale distributional similarity and entity set expansion. In *Proceedings of EMNLP '09*, pages 938–947.

Pennacchiotti, Marco and Patrick Pantel. 2009. Entity extraction via ensemble semantics. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 238–247.

Ravi, Sujith and Marius Pasca. 2008. Using structured text for large-scale attribute extraction. In *CIKM-08*, pages 1183–1192.

Shinzato, Keiji and Kentaro Torisawa. 2004. Extracting hyponyms of prespecified hypernyms from itemizations and headings in web documents. In *Proceedings of COLING '04*, pages 938–944.

Shinzato, Keiji, Tomohide Shibata, Daisuke Kawahara, Chikara Hashimoto, and Sadao Kurohashi. 2008. Tsubaki: An open search engine infrastructure for developing new information access. In *Proceedings of IJCNLP '08*, pages 189–196.

Snow, Rion, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 801–808.

Suchanek, Fabian M., Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A Core of Semantic Knowledge. In *Proc. of WWW '07*, pages 697–706.

Sumida, Asuka and Kentaro Torisawa. 2008. Hacking Wikipedia for hyponymy relation acquisition. In *Proc. of the Third International Joint Conference on Natural Language Processing (IJCNLP)*, pages 883–888, January.

Talukdar, Partha Pratim, Joseph Reisinger, Marius Pasca, Deepak Ravichandran, Rahul Bhagat, and Fernando Pereira. 2008. Weakly-supervised acquisition of labeled class instances using graph random walks. In *Proc. of EMNLP08*, pages 582–590.

TinySVM. 2002. `http://chasen.org/~taku/software/TinySVM`.

Van Durme, Benjamin and Marius Pasca. 2008. Finding cars, goddesses and enzymes: Parametrizable acquisition of labeled instances for open-domain information extraction. In *Proc. of AAAI08*, pages 1243–1248.

Vapnik, Vladimir N. 1995. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA.

Wang, Wei and Zhi-Hua Zhou. 2007. Analyzing co-training style algorithms. In *ECML '07: Proceedings of the 18th European conference on Machine Learning*, pages 454–465.

Simple and Efficient Algorithm for Approximate Dictionary Matching

Naoaki Okazaki
University of Tokyo
okazaki@is.s.u-tokyo.ac.jp

Jun'ichi Tsujii
University of Tokyo
University of Manchester
National Centre for Text Mining
tsujii@is.s.u-tokyo.ac.jp

Abstract

This paper presents a simple and efficient algorithm for approximate dictionary matching designed for similarity measures such as cosine, Dice, Jaccard, and overlap coefficients. We propose this algorithm, called *CPMerge*, for the τ-overlap join of inverted lists. First we show that this task is solvable exactly by a τ-overlap join. Given inverted lists retrieved for a query, the algorithm collects fewer candidate strings and prunes unlikely candidates to efficiently find strings that satisfy the constraint of the τ-overlap join. We conducted experiments of approximate dictionary matching on three large-scale datasets that include person names, biomedical names, and general English words. The algorithm exhibited scalable performance on the datasets. For example, it retrieved strings in 1.1 ms from the string collection of Google Web1T unigrams (with cosine similarity and threshold 0.7).

1 Introduction

Languages are sufficiently flexible to be able to express the same meaning through different diction. At the same time, inconsistency of surface expressions has persisted as a serious problem in natural language processing. For example, in the biomedical domain, *cardiovascular disorder* can be described using various expressions: *cardiovascular diseases*, *cardiovascular system disorder*, and *disorder of the cardiovascular system*. It is a nontrivial task to find the entry from these surface expressions appearing in text.

This paper addresses *approximate dictionary matching*, which consists of finding all strings in a string collection V such that they have similarity that is no smaller than a threshold α with a query string x. This task has a broad range of applications, including spelling correction, flexible dictionary look-up, record linkage, and duplicate detection (Henzinger, 2006; Manku et al., 2007).

Formally, the task obtains a subset $\mathcal{Y}_{x,\alpha} \subseteq V$,

$$\mathcal{Y}_{x,\alpha} = \{y \in V \mid \mathrm{sim}(x, y) \geq \alpha\}, \qquad (1)$$

where $\mathrm{sim}(x, y)$ presents the similarity between x and y. A naïve solution to this task is to compute similarity values $|V|$ times, i.e., between x and every string $y \in V$. However, this solution is impractical when the number of strings $|V|$ is huge (e.g., more than one million).

In this paper, we present a simple and efficient algorithm for approximate dictionary matching designed for similarity measures such as cosine, Dice, Jaccard, and overlap coefficients. Our main contributions are twofold.

1. We show that the problem of approximate dictionary matching is solved exactly by a *τ-overlap join* (Sarawagi and Kirpal, 2004) of inverted lists. Then we present *CPMerge*, which is a simple and efficient algorithm for the τ-overlap join. In addition, the algorithm is easily implemented.

2. We demonstrate the efficiency of the algorithm on three large-scale datasets with person names, biomedical concept names,

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 851–859,
Beijing, August 2010

and general English words. We compare the algorithm with state-of-the-art algorithms, including Locality Sensitive Hashing (Ravichandran et al., 2005; Andoni and Indyk, 2008) and DivideSkip (Li et al., 2008). The proposed algorithm retrieves strings the most rapidly, e.g., in 1.1 ms from Google Web1T unigrams (with cosine similarity and threshold 0.7).

2 Proposed Method

2.1 Necessary and sufficient conditions

In this paper, we assume that the *features* of a string are represented arbitrarily by a set. Although it is important to design a string representation for an accurate similarity measure, we do not address this problem: our emphasis is *not on designing a better representation for string similarity* but *on establishing an efficient algorithm*.

The most popular representation is given by *n-grams*: all substrings of size n in a string. We use trigrams throughout this paper as an example of string representation. For example, the string "methyl sulphone" is expressed by 17 elements of letter trigrams[1], {'$$m', '$me', 'met', 'eth', 'thy', 'hyl', 'yl_', 'l_s', '_su', 'sul', 'ulp', 'lph', 'pho', 'hon', 'one', 'ne$', 'e$$'}. We insert two $s before and after the string to denote the start or end of the string. In general, a string x consisting of $|X|$ letters yields $(|x| + n - 1)$ elements of n-grams. We call $|x|$ and $|X|$ the *length* and *size*, respectively, of the string x.

Let X and Y denote the feature sets of the strings x and y, respectively. The cosine similarity between the two strings x and y is,

$$\text{cosine}(X, Y) = \frac{|X \cap Y|}{\sqrt{|X||Y|}}. \quad (2)$$

By integrating this definition with Equation 1, we obtain the necessary and sufficient condition for approximate dictionary matching,

$$\left\lceil \alpha\sqrt{|X||Y|} \right\rceil \leq |X \cap Y| \leq \min\{|X|, |Y|\}. \quad (3)$$

This inequality states that two strings x and y must have at least $\tau = \left\lceil \alpha\sqrt{|X||Y|} \right\rceil$ features in common. When ignoring $|X \cap Y|$ in the inequality, we have an inequality about $|X|$ and $|Y|$,

$$\lceil \alpha^2 |X| \rceil \leq |Y| \leq \left\lfloor \frac{|X|}{\alpha^2} \right\rfloor \quad (4)$$

This inequality presents the search range for retrieving similar strings; that is, we can ignore strings whose feature size is out of this range. Other derivations are also applicable to similarity measures, including Dice, Jaccard, and overlap coefficients. Table 1 summarizes the conditions for these similarity measures.

Table 1: Conditions for each similarity measure

Measure	$\min \lvert Y \rvert$	$\max \lvert Y \rvert$	$\tau (= \min \lvert X \cap Y \rvert)$
Dice	$\frac{\alpha}{2-\alpha}\lvert X \rvert$	$\frac{2-\alpha}{\alpha}\lvert X \rvert$	$\frac{1}{2}\alpha(\lvert X \rvert + \lvert Y \rvert)$
Jaccard	$\alpha \lvert X \rvert$	$\lvert X \rvert / \alpha$	$\frac{\alpha(\lvert X \rvert + \lvert Y \rvert)}{1+\alpha}$
Cosine	$\alpha^2 \lvert X \rvert$	$\lvert X \rvert / \alpha^2$	$\alpha\sqrt{\lvert X \rvert \lvert Y \rvert}$
Overlap	—	—	$\alpha \min\{\lvert X \rvert, \lvert Y \rvert\}$

We explain one usage of these conditions. Let query string x = "methyl sulphone" and threshold for approximate dictionary matching $\alpha = 0.7$ with cosine similarity. Representing the strings with letter trigrams, we have the size of x, $|X| = 17$. The inequality 4 gives the search range of $|Y|$ of the retrieved strings, $9 \leq |Y| \leq 34$. Presuming that we are searching for strings of $|Y| = 16$, we obtain the necessary and sufficient condition for the approximate dictionary matching from the inequality 3, $\tau = 12 \leq |X \cap Y|$. Thus, we need to search for strings that have at least 12 letter trigrams that overlap with X. When considering a string y = "methyl sulfone", which is a spelling variant of y (*ph* → *f*), we confirm that the string is a solution for approximate dictionary matching because $|X \cap Y| = 13 (\geq \tau)$. Here, the actual similarity is $\text{cosine}(X, Y) = 13/\sqrt{17 \times 16} = 0.788$ $(\geq \alpha)$.

2.2 Data structure and algorithm

Algorithm 1 presents the pseudocode of the approximate dictionary matching based on Table 1.

[1] In practice, we attach ordinal numbers to n-grams to represent multiple occurrences of n-grams in a string (Chaudhuri et al., 2006). For example, the string "prepress", which contains two occurrences of the trigram 'pre', yields the set {'$$p'#1, '$pr'#1, 'pre'#1, 'rep'#1, 'epr'#1, 'pre'#2, 'res'#1, 'ess'#1, 'ss$'#1, 's$$'#1}.

```
Input: V: collection of strings
Input: x: query string
Input: α: threshold for the similarity
Output: Y: list of strings similar to the query
1  X ← string_to_feature(x);
2  Y ←[];
3  for l ← min_y(|X|, α) to max_y(|X|, α) do
4  |   τ ← min_overlap(|X|, l, α);
5  |   R ← overlapjoin(X, τ, V, l);
6  |   foreach r ∈ R do append r to Y;
7  end
8  return Y;
```

Algorithm 1: Approximate dictionary matching.

```
Input: X: array of features of the query string
Input: τ: minimum number of overlaps
Input: V: collection of strings
Input: l: size of target strings
Output: R: list of strings similar to the query
1  M ← {};
2  R ← [];
3  foreach q ∈ X do
4  |   foreach i ∈ get(V, l, q) do
5  |   |   M[i] ← M[i] + 1;
6  |   |   if τ ≤ M[i] then
7  |   |   |   append i to R;
8  |   |   end
9  |   end
10 end
11 return R;
```

Algorithm 2: AllScan algorithm.

Given a query string x, a collection of strings V, and a similarity threshold α, the algorithm computes the size range (line 3) given by Table 1. For each size l in the range, the algorithm computes the minimum number of overlaps τ (line 4). The function `overlapjoin` (line 5) finds similar strings by solving the following problem (τ-*overlap join*): given a list of features of the query string X and the minimum number of overlaps τ, enumerate strings of size l in the collection V such that they have at least τ feature overlaps with X.

To solve this problem efficiently, we build an inverted index that stores a mapping from the features to their originating strings. Then, we can perform the τ-overlap join by finding strings that appear at least τ times in the inverted lists retrieved for the query features X.

Algorithm 2 portrays a naïve solution for the τ-overlap join (AllScan algorithm). In this algorithm, function `get`(V, l, q) returns the inverted list of strings (of size l) for the feature q. In short, this algorithm scans strings in the inverted lists retrieved for the query features X, counts the frequency of occurrences of every string in the inverted lists, and returns the strings whose frequency of occurrences is no smaller than τ.

This algorithm is inefficient in that it scans all strings in the inverted lists. The number of scanned strings is large, especially when some query features appear frequently in the strings, e.g., `'s$$'` (words ending with 's') and `'pre'` (words with substring 'pre'). To make matters worse, such features are too common for characterizing string similarity. The AllScan algorithm is able to maintain numerous candidate strings in M, but most candidates are not likely to qualified because they have few overlaps with X.

To reduce the number of the candidate strings, we refer to *signature-based algorithms* (Arasu et al., 2006; Chaudhuri et al., 2006):

Property 1 *Let there be a set (of size h) X and a set (of any size) Y. Consider any subset $Z \subseteq X$ of size $(h - \tau + 1)$. If $|X \cap Y| \geq \tau$, then $Z \cap Y \neq \phi$.*

We explain one usage of this property. Let query string x = "methyl sulphone" and its trigram set X be features (therefore, $|X| = h = 17$). Presuming that we seek strings whose trigrams are size 16 and have 12 overlaps with X, then string y must have at least one overlap with any subset of size 6 ($= 17 - 12 + 1$) of X. We call the subset *signatures*. The property leads to an algorithmic design by which we obtain a small set of candidate strings from the inverted lists for signatures, $(|X| - \tau + 1)$ features in X, and verify whether each candidate string satisfies the τ overlap with the remaining $(\tau - 1)$ n-grams.

Algorithm 3 presents the pseudocode employing this idea. In line 1, we arrange the features in X in ascending order of the number of strings in their inverted lists. We denote the k-th element in the ordered features as X_k ($k \in \{0, ..., |X| - 1\}$), where the index number begins with 0. Based on this notation, X_0 and $X_{|X|-1}$ are the most uncommon and the most common features in X, respectively.

In lines 2–7, we use $(|X| - \tau + 1)$ features

```
    Input: X: array of features of the query string
    Input: τ: minimum number of overlaps
    Input: V: collection of strings
    Input: l: size of target strings
    Output: R: list of strings similar to the query
 1  sort elements in X by order of |get(V, l, X_k)|;
 2  M ← {};
 3  for k ← 0 to (|X| − τ) do
 4  |   foreach s ∈ get(V, l, X_k) do
 5  |   |   M[s] ← M[s] + 1;
 6  |   end
 7  end
 8  R ← [];
 9  for k ← (|X| − τ + 1) to (|X| − 1) do
10  |   foreach s ∈ M do
11  |   |   if bsearch(get(V, l, X_k), s) then
12  |   |   |   M[s] ← M[s] + 1;
13  |   |   end
14  |   |   if τ ≤ M[s] then
15  |   |   |   append s to R;
16  |   |   |   remove s from M;
17  |   |   else if M[s] + (|X| − k − 1) < τ then
18  |   |   |   remove s from M;
19  |   |   end
20  |   end
21  end
22  return R;
```

Algorithm 3: CPMerge algorithm.

$X_0, ..., X_{|X|-\tau}$ to generate a compact set of candidate strings. The algorithm stores the occurrence count of each string s in $M[s]$. In lines 9–21, we increment the occurrence counts if each of $X_{|X|-\tau+1}, ..., X_{|X|-1}$ inverted lists contain the candidate strings. For each string s in the candidates (line 10), we perform a binary search on the inverted list (line 11), and increment the overlap count if the string s exists (line 12). If the overlap counter of the string reaches τ (line 14), then we append the string s to the result list R and remove s from the candidate list (lines 15–16). We prune a candidate string (lines 17–18) if the candidate is found to be unreachable for τ overlaps even if it appears in all of the unexamined inverted lists.

3 Experiments

We report the experimental results of approximate dictionary matching on large-scale datasets with person names, biomedical names, and general English words. We implemented various systems of approximate dictionary matching.

- **Proposed**: CPMerge algorithm.
- **Naive**: Naïve algorithm that computes the cosine similarity $|V|$ times for every query.
- **AllScan**: AllScan algorithm.
- **Signature**: CPMerge algorithm without pruning; this is equivalent to Algorithm 3 without lines 17–18.
- **DivideSkip**: our implementation of the algorithm (Li et al., 2008)[2].
- **Locality Sensitive Hashing (LSH)** (Andoni and Indyk, 2008): This baseline system follows the design of previous work (Ravichandran et al., 2005). This system *approximately* solves Equation 1 by finding dictionary entries whose LSH values are within the (bit-wise) hamming distance of θ from the LSH value of a query string. To adapt the method to approximate dictionary matching, we used a 64-bit LSH function computed with letter trigrams. By design, this method does not find an exact solution to Equation 1; in other words, the method can miss dictionary entries that are actually similar to the query strings. This system has three parameters, θ, q (number of bit permutations), and B (search width), to control the tradeoff between retrieval speed and recall[3]. Generally speaking, increasing these parameters improves the recall, but slows down the speed. We determined $\theta = 24$ and $q = 24$ experimentally[4], and measured the performance when $B \in \{16, 32, 64\}$.

The systems, excluding LSH, share the same implementation of Algorithm 1 so that we can specifically examine the differences of the algorithms for τ-overlap join. The C++ source code of the system used for this experiment is available[5]. We ran all experiments on an application server running Debian GNU/Linux 4.0 with Intel Xeon 5140 CPU (2.33 GHz) and 8 GB main memory.

[2] We tuned parameter values $\mu \in \{0.01, 0.02, 0.04, 0.1, 0.2, 0.4, 1, 2, 4, 10, 20, 40, 100\}$ for each dataset. We selected the parameter with the fastest response.

[3] We followed the notation of the original paper (Ravichandran et al., 2005) here. Refer to the original paper for definitions of the parameters θ, q, and B.

[4] q was set to 24 so that the arrays of shuffled hash values are stored in memory. We chose $\theta = 24$ from $\{8, 16, 24\}$ because it showed a good balance between accuracy and speed.

[5] http://www.chokkan.org/software/simstring/

3.1 Datasets

We used three large datasets with person names (IMDB actors), general English words (Google Web1T), and biomedical names (UMLS).

- **IMDB actors**: This dataset comprises actor names extracted from the IMDB database[6]. We used all actor names (1,098,022 strings; 18 MB) from the file `actors.list.gz`. The average number of letter trigrams in the strings is 17.2. The total number of trigrams is 42,180. The system generated index files of 83 MB in 56.6 s.
- **Google Web1T unigrams**: This dataset consists of English word unigrams included in the Google Web1T corpus (LDC2006T13). We used all word unigrams (13,588,391 strings; 121 MB) in the corpus after removing the frequency information. The average number of letter trigrams in the strings is 10.3. The total number of trigrams is 301,459. The system generated index files of 601 MB in 551.7 s.
- **UMLS**: This dataset consists of English names and descriptions of biomedical concepts included in the Unified Medical Language System (UMLS). We extracted all English concept names (5,216,323 strings; 212 MB) from `MRCONSO.RRF.aa.gz` and `MRCONSO.RRF.ab.gz` in UMLS Release 2009AA. The average number of letter trigrams in the strings is 43.6. The total number of trigrams is 171,596. The system generated index files of 1.1 GB in 1216.8 s.

For each dataset, we prepared 1,000 query strings by sampling strings randomly from the dataset. To simulate the situation where query strings are not only identical but also similar to dictionary entries, we introduced random noise to the strings. In this experiment, one-third of the query strings are unchanged from the original (sampled) strings, one-third of the query strings have one letter changed, and one-third of the query strings have two letters changed. When changing a letter, we randomly chose a letter position from a uniform distribution, and replaced the letter at the position with an ASCII letter randomly chosen from a uniform distribution.

[6] `ftp://ftp.fu-berlin.de/misc/movies/database/`

3.2 Results

To examine the scalability of each system, we controlled the number of strings to be indexed from 10%–100%, and issued 1,000 queries. Figure 1 portrays the average response time for retrieving strings whose cosine similarity values are no smaller than 0.7. Although LSH (B=16) seems to be the fastest in the graph, this system missed many true positives[7]; the recall scores of approximate dictionary matching were 15.4% (IMDB), 13.7% (Web1T), and 1.5% (UMLS). Increasing the parameter B improves the recall at the expense of the response time. LSH (B=64)[8]. It not only ran slower than the proposed method, but also suffered from low recall scores, 25.8% (IMDB), 18.7% (Web1T), and 7.1% (UMLS). LSH was useful only when we required a quick response much more than recall.

The other systems were guaranteed to find the exact solution (100% recall). The proposed algorithm was the fastest of all exact systems on all datasets: the response times per query (100% index size) were 1.07 ms (IMDB), 1.10 ms (Web1T), and 20.37 ms (UMLS). The response times of the Naïve algorithm were too slow, 32.8 s (IMDB), 236.5 s (Web1T), and 416.3 s (UMLS).

The proposed algorithm achieved substantial improvements over the AllScan algorithm: the proposed method was 65.3 times (IMDB), 227.5 times (Web1T), and 13.7 times (UMLS) faster than the Naïve algorithm. We observed that the Signature algorithm, which is Algorithm 3 without lines 17–18, did not perform well: The Signature algorithm was 1.8 times slower (IMDB), 2.1 times faster (Web1T), and 135.0 times slower (UMLS) than the AllScan algorithm. These results indicate that it is imperative to minimize the number of candidates to reduce the number of binary-search operations. The proposed algorithm was 11.1–13.4 times faster than DivideSkip.

Figure 2 presents the average response time

[7] Solving Equation 1, all systems are expected to retrieve the exact set of strings retrieved by the Naïve algorithm.

[8] The response time of LSH (B=64) on the IMDB dataset was 29.72 ms (100% index size).

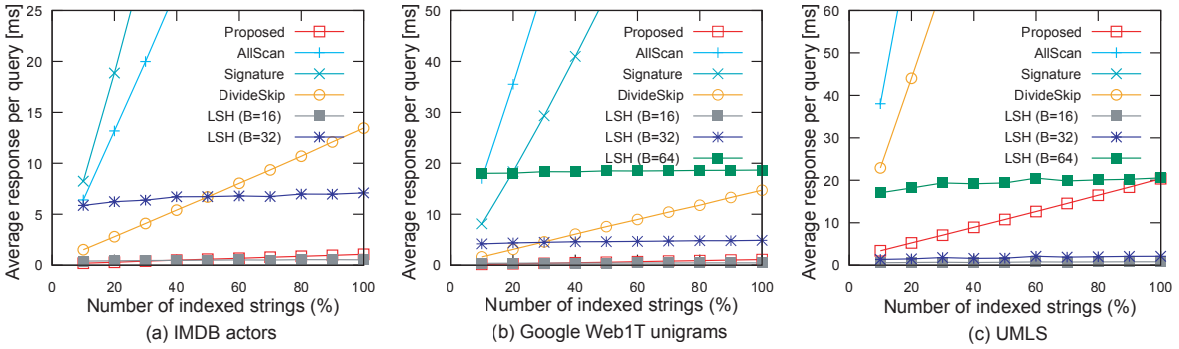


Figure 1: Average response time for processing a query (cosine similarity; $\alpha = 0.7$).

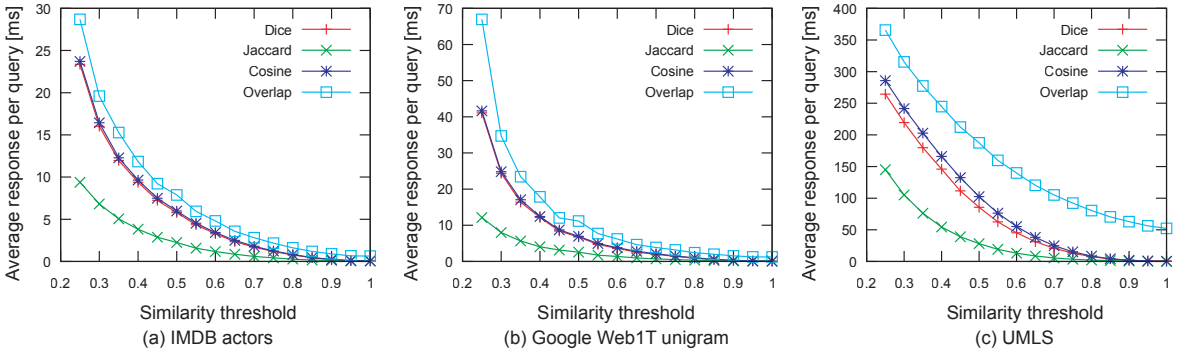


Figure 2: Average response time for processing a query.

of the proposed algorithm for different similarity measures and threshold values. When the similarity threshold is lowered, the algorithm runs slower because the number of retrieved strings $|\mathcal{Y}|$ increases exponentially. The Dice coefficient and cosine similarity produced similar curves.

Table 2 summarizes the run-time statistics of the proposed method for each dataset (with cosine similarity and threshold 0.7). Using the IMDB dataset, the proposed method searched for strings whose size was between 8.74 and 34.06; it retrieved 4.63 strings per query string. The proposed algorithm scanned 279.7 strings in 4.6 inverted lists to obtain 232.5 candidate strings. The algorithm performed a binary search on 4.3 inverted lists containing 7,561.8 strings in all. In contrast, the AllScan algorithm had to scan 16,155.1 strings in 17.7 inverted lists and considered 9,788.7 candidate strings, and found only 4.63 similar strings.

This table clearly demonstrates three key contributions of the proposed algorithm for efficient approximate dictionary matching. First, the proposed algorithm scanned far fewer strings than did the AllScan algorithm. For example, to obtain candidate strings in the IMDB dataset, the proposed algorithm scanned 279.7 strings, whereas the AllScan algorithm scanned 16,155.1 strings. Therefore, the algorithm examined only 1.1%–3.5% of the strings in the entire inverted lists in the three datasets. Second, the proposed algorithm considered far fewer candidates than did the AllScan algorithm: the number of candidate strings considered by the algorithm was 1.2%–6.6% of those considered by the AllScan algorithm. Finally, the proposed algorithm read fewer inverted lists than did the AllScan algorithm. The proposed algorithm actually read 8.9 (IMDB), 6.0 (Web1T), and 31.7 (UMLS) inverted lists during the experiments[9]. These values indicate that the proposed algorithm can solve τ-overlap join problems by checking only 50.3% (IMDB), 53.6% (Web1T), and 51.9% of the total inverted lists re-

[9]These values are 4.6 + 4.3, 3.1 + 2.9, and 14.3 + 17.4.

Table 2: Run-time statistics of the proposed algorithm for each dataset

Averaged item	IMDB	Web1T	UMLS	Description
$\min \lvert y \rvert$	8.74	5.35	21.87	minimum size of trigrams of target strings
$\max \lvert y \rvert$	34.06	20.46	88.48	maximum size of trigrams of target strings
τ	14.13	9.09	47.77	minimum number of overlaps required/sufficient per query
$\lvert \mathcal{Y} \rvert$	4.63	3.22	111.79	number of retrieved strings per query
Total				— averaged for each query and target size:
# inverted lists	17.7	11.2	61.1	number of inverted lists retrieved for a query
# strings	16 155.1	52 557.6	49 561.4	number of strings in the inverted list
# unique strings	9 788.7	44 834.6	17 457.5	number of unique strings in the inverted list
Candidate stage				— averaged for each query and target size:
# inverted lists	4.6	3.1	14.3	number of inverted lists scanned for generating candidates
# strings	279.7	552.7	1 756.3	number of strings scanned for generating candidates
# candidates	232.5	523.7	1 149.7	number of candidates generated for a query
Validation stage				— averaged for each query and target size:
# inverted lists	4.3	2.9	17.4	number of inverted lists examined by binary search for a query
# strings	7 561.8	19 843.6	20 443.7	number of strings targeted by binary search

trieved for queries.

4 Related Work

Numerous studies have addressed approximate dictionary matching. The most popular configuration uses n-grams as a string representation and the edit distance as a similarity measure. Gravano et al. (1998; 2001) presented various filtering strategies, e.g., count filtering, position filtering, and length filtering, to reduce the number of candidates. Kim et al. (2005) proposed two-level n-gram inverted indices (n-Gram/2L) to eliminate the redundancy of position information in n-gram indices. Li et al. (2007) explored the use of variable-length grams (VGRAMs) for improving the query performance. Lee et al. (2007) extended n-grams to include wild cards and developed algorithms based on a replacement semilattice. Xiao et al. (2008) proposed the Ed-Join algorithm, which utilizes mismatching n-grams.

Several studies addressed different paradigms for approximate dictionary matching. Bocek et al. (2007) presented the Fast Similarity Search (FastSS), an enhancement of the neighborhood generation algorithms, in which multiple variants of each string record are stored in a database. Wang et al. (2009) further improved the technique of neighborhood generation by introducing partitioning and prefix pruning. Huynh et al. (2006) developed a solution to the k-mismatch problem in compressed suffix arrays. Liu et al. (2008) stored string records in a trie, and proposed a framework called TITAN. These studies are specialized for the edit distance measure.

A few studies addressed approximate dictionary matching for similarity measures such as cosine and Jaccard similarities. Chaudhuri et al. (2006) proposed the SSJoin operator for similarity joins with several measures including the edit distance and Jaccard similarity. This algorithm first generates *signatures* for strings, finds all pairs of strings whose signatures overlap, and finally outputs the subset of these candidate pairs that satisfy the similarity predicate. Arasu et al. (2006) addressed *signature schemes*, i.e., methodologies for obtaining signatures from strings. They also presented an implementation of the SSJoin operator in SQL. Although we did not implement this algorithm in SQL, it is equivalent to the Signature algorithm in Section 3.

Sarawagi and Kirpal (2004) proposed the MergeOpt algorithm for the τ-overlap join to approximate string matching with overlap, Jaccard, and cosine measures. This algorithm splits inverted lists for a given query A into two groups, S and L, maintains a heap to collect candidate strings on S, and performs a binary search on L to verify the condition of the τ-overlap join for each candidate string. Their subsequent work includes an efficient algorithm for the top-k search of the overlap join (Chandel et al., 2006).

Li et al. (2008) extended this algorithm to the SkipMerge and DivideSkip algorithms. The SkipMerge algorithm uses a heap to compute the τ-overlap join on entire inverted lists A, but has an additional mechanism to increment the fron-

tier pointers of inverted lists efficiently based on the strings popped most recently from the heap. Consequently, SkipMerge can reduce the number of strings that are pushed to the heap. Similarly to the MergeOpt algorithm, DivideSkip splits inverted lists A into two groups S and L, but it applies SkipMerge to S. In Section 3, we reported the performance of DivideSkip.

Charikar (2002) presented the Locality Sensitive Hash (LSH) function (Andoni and Indyk, 2008), which preserves the property of cosine similarity. The essence of this function is to map strings into N-bit hash values where the bitwise hamming distance between the hash values of two strings approximately corresponds to the angle of the two strings. Ravichandran et al. (2005) applied LSH to the task of noun clustering. Adapting this algorithm to approximate dictionary matching, we discussed its performance in Section 3.

Several researchers have presented refined similarity measures for strings (Winkler, 1999; Cohen et al., 2003; Bergsma and Kondrak, 2007; Davis et al., 2007). Although these studies are sometimes regarded as a research topic of approximate dictionary matching, they assume that two strings for the target of similarity computation are given; in other words, it is out of their scope to find strings in a large collection that are similar to a given string. Thus, it is a reasonable approach for an approximate dictionary matching to quickly collect candidate strings with a loose similarity threshold, and for a refined similarity measure to scrutinize each candidate string for the target application.

5 Conclusions

We present a simple and efficient algorithm for approximate dictionary matching with the cosine, Dice, Jaccard, and overlap measures. We conducted experiments of approximate dictionary matching on large-scale datasets with person names, biomedical names, and general English words. Even though the algorithm is very simple, our experimental results showed that the proposed algorithm executed very quickly. We also confirmed that the proposed method drastically reduced the number of candidate strings considered during approximate dictionary matching. We believe that this study will advance practical NLP applications for which the execution time of approximate dictionary matching is critical.

An advantage of the proposed algorithm over existing algorithms (e.g., MergeSkip) is that it does not need to read all the inverted lists retrieved by query n-grams. We observed that the proposed algorithm solved τ-overlap joins by checking approximately half of the inverted lists (with cosine similarity and threshold $\alpha = 0.7$). This characteristic is well suited to processing compressed inverted lists because the algorithm needs to decompress only half of the inverted lists. It is natural to extend this study to compressing and decompressing inverted lists for reducing disk space and for improving query performance (Behm et al., 2009).

Acknowledgments

This work was partially supported by Grants-in-Aid for Scientific Research on Priority Areas (MEXT, Japan) and for Solution-Oriented Research for Science and Technology (JST, Japan).

References

Andoni, Alexandr and Piotr Indyk. 2008. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117–122.

Arasu, Arvind, Venkatesh Ganti, and Raghav Kaushik. 2006. Efficient exact set-similarity joins. In *VLDB '06: Proceedings of the 32nd International Conference on Very Large Data Bases*, pages 918–929.

Behm, Alexander, Shengyue Ji, Chen Li, and Jiaheng Lu. 2009. Space-constrained gram-based indexing for efficient approximate string search. In *ICDE '09: Proceedings of the 2009 IEEE International Conference on Data Engineering*, pages 604–615.

Bergsma, Shane and Grzegorz Kondrak. 2007. Alignment-based discriminative string similarity. In *ACL '07: Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 656–663.

Bocek, Thomas, Ela Hunt, and Burkhard Stiller. 2007. Fast similarity search in large dictionaries. Technical Report ifi-2007.02, Department of Informatics (IFI), University of Zurich.

Chandel, Amit, P. C. Nagesh, and Sunita Sarawagi. 2006. Efficient batch top-k search for dictionary-based entity recognition. In *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering*.

Charikar, Moses S. 2002. Similarity estimation techniques from rounding algorithms. In *STOC '02: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388.

Chaudhuri, Surajit, Venkatesh Ganti, and Raghav Kaushik. 2006. A primitive operator for similarity joins in data cleaning. In *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering*.

Cohen, William W., Pradeep Ravikumar, and Stephen E. Fienberg. 2003. A comparison of string distance metrics for name-matching tasks. In *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web (IIWeb-03)*, pages 73–78.

Davis, Jason V., Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S. Dhillon. 2007. Information-theoretic metric learning. In *ICML '07: Proceedings of the 24th International Conference on Machine Learning*, pages 209–216.

Gravano, Luis, Panagiotis G. Ipeirotis, H. V. Jagadish, Nick Koudas, S. Muthukrishnan, and Divesh Srivastava. 2001. Approximate string joins in a database (almost) for free. In *VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases*, pages 491–500.

Henzinger, Monika. 2006. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In *SIGIR '06: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 284–291.

Huynh, Trinh N. D., Wing-Kai Hon, Tak-Wah Lam, and Wing-Kin Sung. 2006. Approximate string matching using compressed suffix arrays. *Theoretical Computer Science*, 352(1-3):240–249.

Kim, Min-Soo, Kyu-Young Whang, Jae-Gil Lee, and Min-Jae Lee. 2005. n-Gram/2L: a space and time efficient two-level n-gram inverted index structure. In *VLDB '05: Proceedings of the 31st International Conference on Very Large Data Bases*, pages 325–336.

Lee, Hongrae, Raymond T. Ng, and Kyuseok Shim. 2007. Extending q-grams to estimate selectivity of string matching with low edit distance. In *VLDB '07: Proceedings of the 33rd International Conference on Very Large Data Bases*, pages 195–206.

Li, Chen, Bin Wang, and Xiaochun Yang. 2007. Vgram: improving performance of approximate queries on string collections using variable-length grams. In *VLDB '07: Proceedings of the 33rd International Conference on Very Large Data Bases*, pages 303–314.

Li, Chen, Jiaheng Lu, and Yiming Lu. 2008. Efficient merging and filtering algorithms for approximate string searches. In *ICDE '08: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, pages 257–266.

Liu, Xuhui, Guoliang Li, Jianhua Feng, and Lizhu Zhou. 2008. Effective indices for efficient approximate string search and similarity join. In *WAIM '08: Proceedings of the 2008 The Ninth International Conference on Web-Age Information Management*, pages 127–134.

Manku, Gurmeet Singh, Arvind Jain, and Anish Das Sarma. 2007. Detecting near-duplicates for web crawling. In *WWW '07: Proceedings of the 16th International Conference on World Wide Web*, pages 141–150.

Navarro, Gonzalo and Ricardo Baeza-Yates. 1998. A practical q-gram index for text retrieval allowing errors. *CLEI Electronic Journal*, 1(2).

Ravichandran, Deepak, Patrick Pantel, and Eduard Hovy. 2005. Randomized algorithms and nlp: using locality sensitive hash function for high speed noun clustering. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 622–629.

Sarawagi, Sunita and Alok Kirpal. 2004. Efficient set joins on similarity predicates. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 743–754.

Wang, Wei, Chuan Xiao, Xuemin Lin, and Chengqi Zhang. 2009. Efficient approximate entity extraction with edit distance constraints. In *SIGMOD '09: Proceedings of the 35th SIGMOD International Conference on Management of Data*, pages 759–770.

Winkler, William E. 1999. The state of record linkage and current research problems. Technical Report R99/04, Statistics of Income Division, Internal Revenue Service Publication.

Xiao, Chuan, Wei Wang, and Xuemin Lin. 2008. Ed-Join: an efficient algorithm for similarity joins with edit distance constraints. In *VLDB '08: Proceedings of the 34th International Conference on Very Large Data Bases*, pages 933–944.

Latent Mixture of Discriminative Experts for Multimodal Prediction Modeling

Derya Ozkan, Kenji Sagae and Louis-Philippe Morency
USC Institute for Creative Technologies
{ozkan,sagae,morency}@ict.usc.edu

Abstract

During face-to-face conversation, people naturally integrate speech, gestures and higher level language interpretations to predict the right time to start talking or to give backchannel feedback. In this paper we introduce a new model called Latent Mixture of Discriminative Experts which addresses some of the key issues with multimodal language processing: (1) temporal synchrony/asynchrony between modalities, (2) micro dynamics and (3) integration of different levels of interpretation. We present an empirical evaluation on listener nonverbal feedback prediction (e.g., head nod), based on observable behaviors of the speaker. We confirm the importance of combining four types of multimodal features: lexical, syntactic structure, eye gaze, and prosody. We show that our Latent Mixture of Discriminative Experts model outperforms previous approaches based on Conditional Random Fields (CRFs) and Latent-Dynamic CRFs.

1 Introduction

Face-to-face communication is highly interactive. Even when only one person speaks at a time, other participants exchange information continuously amongst themselves and with the speaker through gestures, gaze and prosody. These different channels contain complementary information essential to interpretation and understanding of human behaviors (Oviatt, 1999). Psycholinguistic studies also suggest that gesture and speech come from a single underlying mental process, and they are related both temporally and semantically (McNeill, 1992; Cassell and Stone, 1999; Kendon, 2004).

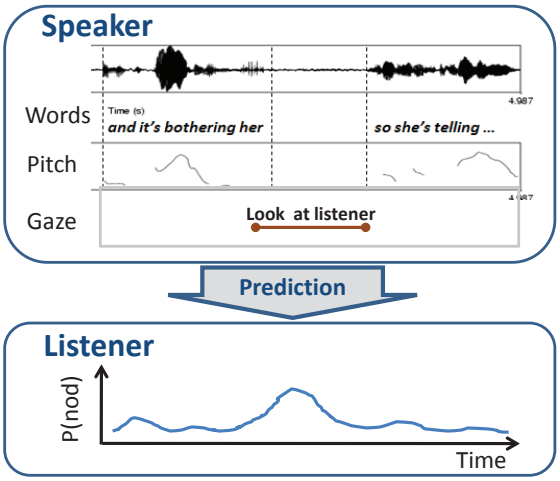


Figure 1: Example of multimodal prediction model: listener nonverbal backchannel prediction based on speaker's speech and eye gaze. As the speaker says the word *her*, which is the end of the clause (*her* is also the object of the verb *bothering*), and lowers the pitch while looking back at the listener and eventually pausing, the listener is then very likely to head nod (i.e., nonverbal backchannel).

A good example of such complementarity is how people naturally integrate speech, gestures and higher level language to predict when to give backchannel feedback. Building computational models of such a predictive process is challenging since it involves micro dynamics and temporal relationship between cues from different modalities (Quek, 2003). Figure 1 shows an example of backchannel prediction where a listener head nod

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 860–868,
Beijing, August 2010

is more likely. For example, a temporal sequence from the speaker where he/she reaches the end of segment (syntactic feature) with a low pitch and looks at the listener before pausing is a good opportunity for the listener to give nonverbal feedback (e.g., head nod). These prediction models have broad applicability, including the improvement of nonverbal behavior recognition, the synthesis of natural animations for robots and virtual humans, the training of cultural-specific nonverbal behaviors, and the diagnoses of social disorders (e.g., autism spectrum disorder).

In this paper we introduce a new model called Latent Mixture of Discriminative Experts (LMDE) which addresses some of the key issues with multimodal language processing: (1) temporal synchrony/asynchrony between modalities, (2) micro dynamics and (3) integration of different levels of interpretation. We present an empirical evaluation on nonverbal feedback prediction (e.g., head nod) confirming the importance of combining different types of multimodal features. We show that our LMDE model outperforms previous approaches based Conditional Random Fields (CRFs) and Latent-Dynamic CRFs.

2 Related Work

Earlier work in multimodal language processing focused on multimodal dialogue systems where the gestures and speech may be constrained (Johnston, 1998; Jurafsky et al., 1998). Most of the research in multimodal language processing over the past decade fits within two main trends that have emerged: (1) recognition of individual multimodal actions such as speech and gestures (e.g, (Eisenstein et al., 2008; Frampton et al., 2009; Gravano et al., 2007)), and (2) recognition/summarization of the social interaction between more than one participants (e.g., meeting analysis (Heylen and op den Akker, 2007; Moore, 2007; Murray and Carenini, 2009; Jovanovic et al., 2006)).

The work described in this paper can be seen from a third intermediate category where multimodal cues from one person is used to predict the social behavior of another participant. This type of predictive models has been mostly studied in the context of embodied conversational agents (Nakano et al., 2003; Nakano et al., 2007). In particular, backchannel feedback (the nods and paraverbals such as "uh-hu" and "mm-hmm" that listeners produce as someone is speaking) has received considerable interest due to its pervasiveness across languages and conversational contexts and this paper addresses the problem of how to predict and generate this important class of dyadic nonverbal behavior.

Several researchers have developed models to predict when backchannel should happen. In general, these results are difficult to compare as they utilize different corpora and present varying evaluation metrics. Ward and Tsukahara (2000) propose a unimodal approach where backchannels are associated with a region of low pitch lasting 110ms during speech. Models were produced manually through an analysis of English and Japanese conversational data. Nishimura et al. (2007) present a unimodal decision-tree approach for producing backchannels based on prosodic features. Cathcart et al. (2003) propose a unimodal model based on pause duration and trigram part-of-speech frequency. The model was constructed by identifying, from the HCRC Map Task Corpus (Anderson et al., 1991), trigrams ending with a backchannel. Fujie et al. (2004) used Hidden Markov Models to perform head nod recognition. In their paper, they combined head gesture detection with prosodic low-level features from the same person to determine strongly positive, weak positive and negative responses to yes/no type utterances.

In recent years, great research has shown the strength of latent variable models for natural language processing (Blunsom et al., 2008). One of the most relevant works is that of Eisenstein and Davis (2007), which presents a latent conditional model for fusion of multiple modalities (speech and gestures). One of the key difference of our work is that we are explicitly modeling the micro dynamics and temporal relationship between modalities.

3 Multimodal Prediction Models

Human face-to-face communication is a little like a dance, in that participants continuously adjust their behaviors based on verbal and nonverbal dis-

plays and signals. A topic of central interest in modeling such behaviors is the patterning of interlocutor actions and interactions, moment-by-moment, and one of the key challenges is identifying the patterns that best predict specific actions. Thus we are interested in developing predictive models of communication dynamics that integrate previous and current actions from all interlocutors to anticipate the most likely next actions of one or all interlocutors. Humans are good at this: they have an amazing ability to predict, at a micro-level, the actions of an interlocutor (Bavelas et al., 2000); and we know that better predictions can correlate with more empathy and better outcomes (Goldberg, 2005; Fuchs, 1987).

With turn-taking being perhaps the best-known example, we now know a fair amount about some aspects of communication dynamics, but much less about others. However, recent advances in machine learning and experimental methods, and recent findings from a variety of perspectives, including conversation analysis, social signal processing, adaptation, corpus analysis and modeling, perceptual experiments, and dialog systems-building and experimentation, mean that the time is ripe to start working towards more comprehensive predictive models.

The study of multimodal prediction models bring a new series of research challenges:

MULTIMODAL ASYNCHRONY While speech and gestures seem to come from a single underlying mental process (McNeill, 1992), they not always happen at the same time, making it hard for earlier multimodal fusion approaches based on synchrony. A multimodal prediction model needs to be able to learn automatically the temporal relationship (and relative importance) between modalities.

MICRO DYNAMICS The dynamic between multimodal signals should be taken at a micro level since many of the interactions between speech and gesture happen at the sub-gesture level or sub-word level (Quek, 2003). Typical word-based sampling may not be sufficient and instead a higher sampling rate should be used.

LIMITED ANNOTATED DATA Given the time requirement to correctly annotate multimodal data, most multimodal datasets contain only a limited number of labeled examples. Since many machine learning algorithms rely on a large training corpus, effective training of a predictive model on multimodal datasets is challenging.

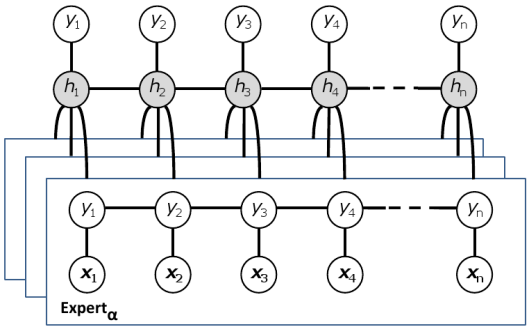


Figure 2: Latent Mixture of Discriminative Experts: a new dynamic model for multimodal fusion. In this graphical model, x_j represents the j^{th} multimodal observation, h_j is a hidden state assigned to x_j , and y_j the class label of x_j. Gray circles are latent variables. The micro dynamics and multimodal temporal relationships are automatically learned by the hidden states h_j during the learning phase.

4 Latent Mixture of Discriminative Experts

In this paper we present a multimodal fusion algorithm, called Latent Mixture of Discriminative Experts (shown in Figure 2), that addresses the three challenges discussed in the previous section. The hidden states of LMDE automatically learn the temporal asynchrony between modalities. By using a constant sample rate of 30Hz in our experiments, we can model the micro dynamics of speech and prosody (e.g., change of intonation in the middle of a word). And finally, by training separate experts for each modalities, we improve the prediction performance even with limited datasets.

The task of our LMDE model is to learn a mapping between a sequence of multimodal observations $\mathbf{x} = \{x_1, x_2, ..., x_m\}$ and a sequence of labels $\mathbf{y} = \{y_1, y_2, ..., y_m\}$. Each y_j is a class label for the j^{th} frame of a video sequence and is a member of a set $\mathcal{Y}$ of possible class labels, for example, $\mathcal{Y} = \{\texttt{head-nod}, \texttt{other-gesture}\}$.

Each frame observation x_j is represented by a feature vector $\phi(x_j) \in \mathbf{R}^d$, for example, the prosodic features at each sample. For each sequence, we also assume a vector of "sub-structure" variables $\mathbf{h} = \{h_1, h_2, ..., h_m\}$. These variables are not observed in the training examples and will therefore form a set of hidden variables in the model.

Following Morency et al. (2007), we define our LMDE model as follows:

$$P(\mathbf{y} \mid \mathbf{x}, \theta) = \sum_{\mathbf{h}} P(\mathbf{y} \mid \mathbf{h}, \mathbf{x}, \theta) P(\mathbf{h} \mid \mathbf{x}, \theta) \quad (1)$$

where θ is the model parameters that is to be estimated from training data.

To keep training and inference tractable, Morency et al. (2007) restrict the model to have disjoint sets of hidden states associated with each class label. Each h_j is a member of a set $\mathcal{H}_{y_j}$ of possible hidden states for the class label y_j. $\mathcal{H}$, the set of all possible hidden states, is defined to be the union of all $\mathcal{H}_y$ sets. Since sequences which have any $h_j \notin \mathcal{H}_{y_j}$ will by definition have $P(\mathbf{y} \mid \mathbf{h}, \mathbf{x}, \theta) = 0$, latent conditional model becomes:

$$P(\mathbf{y} \mid \mathbf{x}, \theta) = \sum_{\mathbf{h}:\forall h_j \in \mathcal{H}_{y_j}} P(\mathbf{h} \mid \mathbf{x}, \theta). \quad (2)$$

What differentiates our LMDE model from the original work of Morency et al. is the definition of $P(\mathbf{h}|\mathbf{x}, \theta)$:

$$P(\mathbf{h}| \mathbf{x}, \theta) = \frac{\exp\left(\begin{array}{c} \sum_l \theta_l \cdot \mathbf{T}_l(\mathbf{h}, \mathbf{x}) + \\ \sum_\alpha \theta_\alpha \cdot P_\alpha(\mathbf{y}|\mathbf{x}, \lambda_\alpha) \end{array} \right)}{\mathcal{Z}(\mathbf{x}, \theta)}, \quad (3)$$

where $\mathcal{Z}$ is the partition function and $P_\alpha(\mathbf{y}|\mathbf{x})$ is the conditional distribution of the expert indexed by α. The expert conditional distributions are defined $P_\alpha(\mathbf{y}|\mathbf{x}, \lambda_\alpha)$ using the usual conditional random field formulation:

$$P_\alpha(\mathbf{y}| \mathbf{x}, \lambda_\alpha) = \frac{\exp\left(\sum_k \lambda_{\alpha,k} \cdot \mathbf{F}_{\alpha,k}(\mathbf{y}, \mathbf{x})\right)}{\mathcal{Z}_\alpha(\mathbf{x}, \lambda_\alpha)}, \quad (4)$$

$F_{\alpha,k}$ is defined as

$$\mathbf{F}_{\alpha,k}(\mathbf{y}, \mathbf{x}) = \sum_{j=1}^{m} f_{\alpha,k}(y_{j-1}, y_j, \mathbf{x}, j),$$

and each feature function $f_{\alpha,k}(y_{j-1}, y_j, \mathbf{x}, j)$ is either a state function $s_k(y_j, \mathbf{x}, j)$ or a transition function $t_k(y_{j-1}, y_j, \mathbf{x}, j)$. State functions s_k depend on a single hidden variable in the model while transition functions t_k can depend on pairs of hidden variables. $\mathbf{T}_l(\mathbf{h}, \mathbf{x})$, defined in Equation 3, is a special case, summing only over the transition feature functions $t_l(h_{l-1}, h_l, \mathbf{x}, l)$. Each expert α contains a different subset of $f_{\alpha,k}(y_{j-1}, y_j, \mathbf{x}, j)$. These feature functions are defined in Section 5.2.

4.1 Learning Model Parameters

Given a training set consisting of n labeled sequences $(\mathbf{x_i}, \mathbf{y_i})$ for $i = 1...n$, training is done in a two step process. First each expert α is trained following (Kumar and Herbert., 2003; Lafferty et al., 2001) objective function to learn the parameter λ^*_α:

$$L(\lambda_\alpha) = \sum_{i=1}^{n} \log P_\alpha(\mathbf{y}_i \mid \mathbf{x}_i, \lambda_\alpha) - \frac{1}{2\sigma^2}||\lambda_\alpha||^2 \quad (5)$$

The first term in Eq. 5 is the conditional log-likelihood of the training data. The second term is the log of a Gaussian prior with variance σ^2, i.e., $P(\lambda_\alpha) \sim \exp\left(\frac{1}{2\sigma^2}||\lambda_\alpha||^2\right)$.

Then the marginal probabilities $P_\alpha(y_j = a \mid \mathbf{y}, \mathbf{x}, \lambda^*_\alpha)$, are computed using belief propagation and used as input for Equation 3. The optimal parameter θ^* was learned using the log-likelyhood of the conditional probability defined in Equation 2 (i.e., no regularization).

4.2 Inference

For testing, given a new test sequence $\mathbf{x}$, we want to estimate the most probable sequence of labels $\mathbf{y}^*$ that maximizes our LMDE model:

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} \sum_{\mathbf{h}:\forall h_i \in \mathcal{H}_{y_i}} P(\mathbf{h} \mid \mathbf{x}, \theta^*) \quad (6)$$

5 Experimental Setup

We evaluate our Latent Mixture of Discriminative Experts on the multimodal task of predicting listener nonverbal backchannel (i.e., head nods). Backchannel feedback (the nods and paraverbals such as "uh-hu" and "mm-hmm" that listeners

produce as some is speaking) has received considerable interest due to its pervasiveness across languages and conversational contexts.

5.1 Dataset

We are using the RAPPORT dataset from (Maatman et al., 2005), which contains 47 dyadic interactions between a speaker and a listener. Data is drawn from a study of face-to-face narrative discourse ("quasi-monologic" storytelling). In this dataset, participants in groups of two were told they were participating in a study to evaluate a communicative technology. Subjects were randomly assigned the role of speaker and listener. The speaker viewed a short segment of a video clip taken from the Edge Training Systems, Inc. Sexual Harassment Awareness video. After the speaker finished viewing the video, the listener was led back into the computer room, where the speaker was instructed to retell the stories portrayed in the clips to the listener. The listener was asked to not talk during the story retelling. Elicited stories were approximately two minutes in length on average. Participants sat approximately 8 feet apart. Video sequences were manually annotated to determine the ground truth head nod labels. A total of 587 head nods occured over all video sequences.

5.2 Multimodal Features

This section describes the different multimodal features used to create our five experts.

PROSODY Prosody refers to the rhythm, pitch and intonation of speech. Several studies have demonstrated that listener feedback is correlated with a speaker's prosody (Nishimura et al., 2007; Ward and Tsukahara, 2000; Cathcart et al., 2003). For example, Ward and Tsukahara (2000) show that short listener backchannels (listener utterances like "ok" or "uh-huh" given during a speaker's utterance) are associated with a lowering of pitch over some interval. Listener feedback often follows speaker pauses or filled pauses such as "um" (see (Cathcart et al., 2003)). Using openSMILE (Eyben et al., 2009) toolbox, we extract the following prosodic features, including standard linguistic annotations and the prosodic features suggested by Ward and Tsukhara: downslopes in pitch continuing for at least 40ms, regions of pitch lower than the 26th percentile continuing for at least 110ms (i.e., lowness), drop or rise in energy of speech (i.e., energy edge), Fast drop or rise in energy of speech (i.e., energy fast edge), vowel volume (i.e., vowels are usually spoken softer) and Pause in speech (i.e., no speech).

VISUAL GESTURES Gestures performed by the speaker are often correlated with listener feedback (Burgoon et al., 1995). Eye gaze, in particular, has often been implicated as eliciting listener feedback. Thus, we manually annotate the following contextual feature: speaker looking at the listener.

LEXICAL Some studies have suggested an association between lexical features and listener feedback (Cathcart et al., 2003). Using the transcriptions, we included all individual words (i.e., unigrams) spoken by the speaker during the interactions.

SYNTACTIC STRUCTURE Finally, we attempt to capture syntactic information that may provide relevant cues by extracting four types of features from a syntactic dependency structure corresponding to the utterance. The syntactic structure is produced automatically using a CRF part-of-speech (POS) tagger and a data-driven left-to-right shift-reduce dependency parser (Sagae and Tsujii, 2007), both trained on POS tags and dependency trees extracted from the Switchboard section of the Penn Treebank (Marcus et al., 1994), converted to dependency trees using the Penn2Malt tool[1]. The four syntactic features are:

- Part-of-speech tags for each word (e.g. noun, verb, etc.), taken from the output of the POS tagger
- Grammatical function for each word (e.g. subject, object, etc.), taken directly from the dependency labels produced by the parser
- Part-of-speech of the syntactic head of each word, taken from the dependency links produced by the parser
- Distance and direction from each word to its syntactic head, computed from the dependency links produced by the parser

[1] http://w3.msi.vxu.se/ nivre/research/Penn2Malt.html

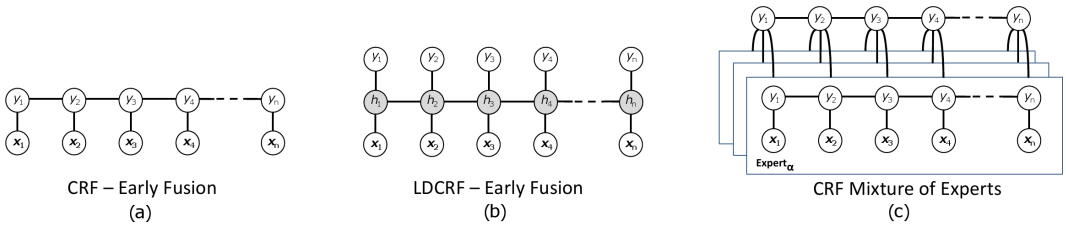


Figure 3: Baseline Models: **a)** Conditional Random Fields (CRF), **b)** Latent Dynamic Conditional Random Fields(LDCRF), **c)** CRF Mixture of Experts (no latent variable)

Although our current method for extracting these features requires that the entire utterance be available for processing, this provides us with a first step towards integrating information about syntactic structure in multimodal prediction models. Many of these features could in principle be computed incrementally with only a slight degradation in accuracy, with the exception of features that require dependency links where a word's syntactic head is to the right of the word itself. We leave an investigation that examines only syntactic features that can be produced incrementally in real time as future work.

5.3 Baseline Models

INDIVIDUAL EXPERTS Our first baseline model consists of a set of CRF chain models, each trained with different set of multimodel features (as described in the previous section). In other words, only visual, prosodic, lexical or syntactic features are used to train a single CRF expert. In one CRF chain model, each gesture class corresponds to a state label. (See Figure 3a).

MULTIMODAL CLASSIFIERS (EARLY FUSION) Our second baseline consists of two models: CRF and LDCRF (Morency et al., 2007). To train these models, we concatenate all multimodal features (lexical, syntactic, prosodic and visual) in one input vector. Graphical representation of these baseline models are given in Figure 3.

CRF MIXTURE OF EXPERTS To show the importance of latent variable in our LMDE model, we trained a CRF-based mixture of discriminative experts. This model is similar to the Logarithmic Opinion Pool (LOP) CRF suggested by Smith et al. (2005). The training is performed in two steps. A graphical representation of a CRF Mixture of experts is given in the last graph of Figure 3.

5.4 Methodology

We performed held-out testing by randomly selecting a subset of 11 interactions (out of 47) for the test set. The training set contains the remaining 36 dyadic interactions. All models in this paper were evaluated with the same training and test sets. Validation of all model parameters (regularization term and number of hidden states) was performed using a 3-fold cross-validation strategy on the training set. The regularization term was validated with values $10^k, k = -1..3$. Three different number of hidden states were tested for the LMDE models: 2, 3 and 4.

The performance is measured by using the F-measure. This is the weighted harmonic mean of precision and recall. Precision is the probability that predicted backchannels correspond to actual listener behavior. Recall is the probability that a backchannel produced by a listener in our test set was predicted by the model. We use the same weight for both precision and recall, so-called F_1. During validation we find all the peaks (i.e., local maxima) from the marginal probabilities. These backchannel hypotheses are filtered using the optimal threshold from the validation set. A backchannel (i.e., head nod) is predicted correctly if a peak happens during an actual listener backchannel with high enough probability. The same evaluation measurement is applied to all models.

The training of all CRFs and LDCRFs were done using the hCRF library[2]. The LMDE model was implemented in Matlab[3] based on the hCRF

[2]http://sourceforge.net/projects/hrcf/

[3]The source code is available at: http://projects.ict.usc.edu/multicomp/.

Table 1: Comparison of individual experts with our Latent Mixture of Discriminative Experts (LMDE).

Expert	Precision	Recall	f1
Lexical	0.1647	0.3305	0.2198
Prosody	0.1396	0.9112	0.2421
Syntactic	0.1833	0.4663	0.2632
POS	0.1935	0.4514	0.2709
Eye Gaze	0.1573	0.1741	0.1653
LMDE	0.2295	0.5677	**0.3268**

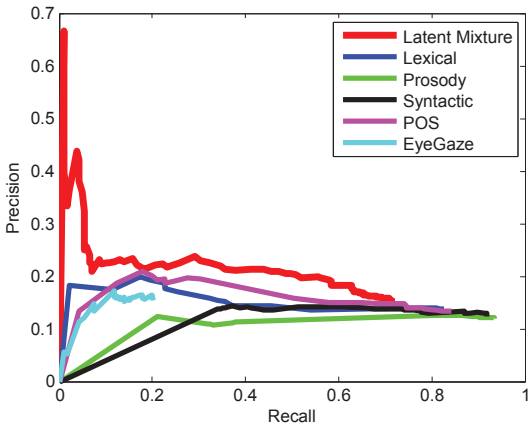


Figure 4: Comparison of individual experts with our LMDE model.

library.

6 Results and Discussion

In this section we present the results of our empirical evaluation designed to test the three main characteristics of the LMDE model: (1) integration of multiple sources of information, (2) late fusion approach and (3) latent variable which models the hidden dynamic between experts. We also present an analysis of the output probabilities from the LMDE model and individual experts.

Individual Experts We trained one individual expert for each feature types: visual, prosodic, lexical and syntactic features (both part-of speech and syntactic structure). Precision, recall and F_1 values for each individual expert and our LMDE model are shown in Table 1 and Figure 4.

Pairwise two-tailed t-test comparison between our LMDE model and individual experts shows a significant difference for Lexical, Prosody, Syntactic and Eye gaze, with respective p-values of 0.0037, 0.0379, 0.0400 and 0.0233. Even though some experts may not perform well individually (e.g., eye gaze), they can bring important information once merged with others. Table 1 shows that our LMDE model was able to take advantage of the complementary information from each expert.

Table 2: Comparison of our Latent Mixture of Discriminative Experts (LMDE) with two early fusion technique (CRF vs LDCRF) and the CRF Mixture of Experts (Smith et al., 2005).

model	Precision	Recall	f1
LMDE	0.2295	0.5677	**0.3268**
Early CRF	0.13958	0.9245	0.2425
Early LDCRF	0.1826	0.2484	0.2105
Mixture CRF	0.1502	0.2712	0.1934

Late Fusion We compare our approach with two early fusion models: CRF and Latent-dynamic CRF (see Figure 3). Table 2 summarizes the results. The CRF model learns direct weights between input features and the gesture labels. The LDCRF is able to model more complex dynamics between input features with the latent variable. We can see that our LMDE model outperforms both early fusion approaches because of its late fusion approach. Pairwise two-tailed t-test analysis gives p-values of 0.0481 and 0.0748, for CRF and LDCRF respectively.

Latent Variable The CRF Mixture of Experts (2005) directly merges the expert outputs while our model uses a latent variable to model the hidden dynamic between experts (see Figure 3). Table 2 summarizes the results. Pairwise two-tailed t-test comparison between these two models shows a significant difference with a p-value of 0.0062. This result is important since it shows that our LMDE model does learn the hidden interaction between experts.

Model analysis To understand the multimodal integration which happens at the latent variable level in our LMDE model, Figure 5 shows the output probabilities for all five individual experts as well as our model. The strength of the latent variable is to enable different weigting

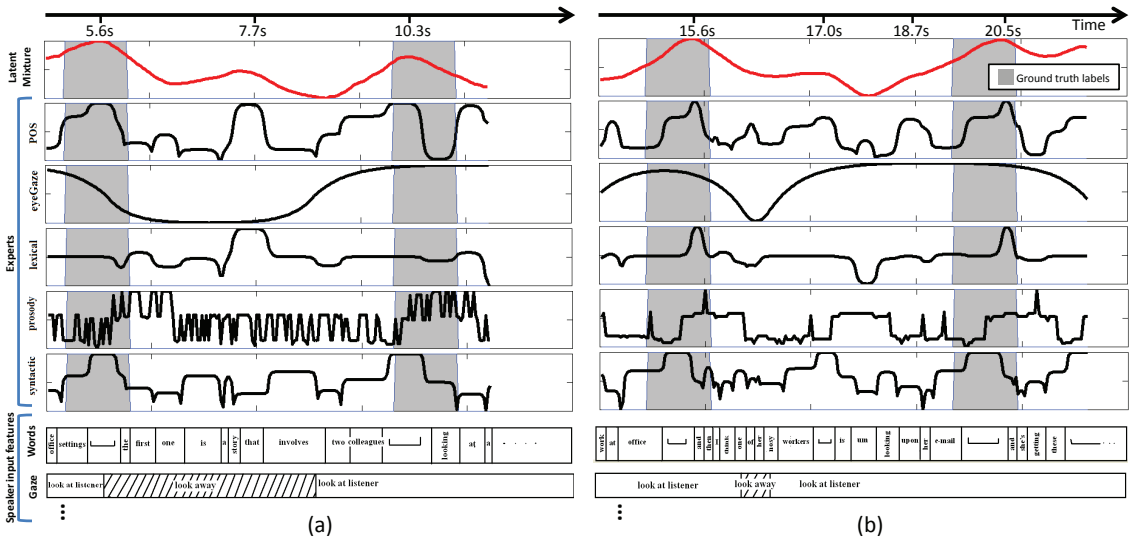


Figure 5: Output probabilities from LMDE and individual experts for two different sub-sequences. The gray areas in the graph corresponds to ground truth backchannel feedbacks of the listener.

of the experts at different point in time.

By analyzing the sequence (a), we observe that both the POS and Syntactic experts learned that when no words are present (i.e., pause) there is a high likelihood of backchennel feedback from the listener (shown at 5.6s and 10.3s). These two experts are highly weighted (by one of the hidden state) during this part of the sequence. Also, both the Lexical and POS experts learned that the word "'that"' (and its part-of-speech) are important but since the speaker is not looking at the listener when saying it, the output from LMDE model is low (see Figure 5, Sequence (a), 7.7s).

By analyzing sequence (b), we see that the Lexical and POS experts learned the importance of the "'and"' at 15.6s and 20.5s. More importantly, we can see at 17.0s and 18.7s that the influence of the POS and Syntactic experts have been reduced in the LMDE output probability. This difference of weighting shows that a different hidden state is active during Sequence (b).

7 Conclusion

In this paper we introduced a new model called Latent Mixture of Discriminative Experts (LMDE) for learning predictive models of human communication behaviors. Many of the interactions between speech and gesture happen at the sub-gesture or sub-word level. LMDE learns automatically the temporal relationship between different modalities. Since, we train separate experts for each modality, LMDE is capable of improving the prediction performance even with limited datasets.

We evaluated our model on the task of nonverbal feedback prediction (e.g., head nod). Our experiments confirm the importance of combining the four types of multimodal features: lexical, syntactic structure, eye gaze, and prosody. LMDE is a generic model that can be applied to a wide range of problems. As future work, we are planning to test our model on dialog act classification and multimodal behavior recognition tasks.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 0917321 and the U.S. Army Research, Development, and Engineering Command (RDECOM). The content does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

References

Anderson, H., M. Bader, E.G. Bard, G. Doherty, S. Garrod, S. Isard, J. Kowtko, J. McAllister, J. Miller, C. Sotillo,

H. Thompson, and R. Weinert. 1991. The mcrc map task corpus. *Language and Speech*, 34(4):351–366.

Bavelas, J.B., L. Coates, and T. Johnson. 2000. Listeners as co-narrators. *JPSP*, 79(6):941–952.

Blunsom, P., T. Cohn, and M. Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *ACL: HLT*, pages 200–208.

Burgoon, Judee K., Lesa A. Stern, and Leesa Dillman. 1995. *Interpersonal adaptation: Dyadic interaction patterns*. Cambridge University Press, Cambridge.

Cassell, J. and M. Stone. 1999. Living hand to mouth: Psychological theories about speech and gesture in interactive dialogue systems. In *AAAI*.

Cathcart, N., Jean Carletta, and Ewan Klein. 2003. A shallow model of backchannel continuers in spoken dialogue. In *EACL*, pages 51–58.

Eisenstein, J., R. Barzilay, and R. Davis. 2008. Gestural cohesion for topic segmentation. In *ACL: HLT*, pages 852–860.

Eisentein, J. and R. Davis. 2007. Conditional modality fusion for coreference. In *ACL*, pages 352–359.

Eyben, Florian, Martin Wöllmer, and Björn Schuller. 2009. openEAR - Introducing the Munich Open-Source Emotion and Affect Recognition Toolkit. In *ACII*, pages 576–581.

Frampton, M., J. Huang, T. Bui, and S. Peters. 2009. Real-time decision detection in multi-party dialogue. In *EMNLP*, pages 1133–1141.

Fuchs, D. 1987. Examiner familiarity effects on test performance: implications for training and practice. *Topics in Early Childhood Special Education*, 7:90–104.

Fujie, Shinya, Yasuhi Ejiri, Kei Nakajima, Yosuke Matsusaka, and Tetsunori Kobayashi. 2004. A conversation robot using head gesture recognition as para-linguistic information. In *RO-MAN*, pages 159–164.

Goldberg, S.B. 2005. The secrets of successful mediators. *Negotiation Journal*, 21(3):365–376.

Gravano, A., S. Benus, H. Chavez, J. Hirschberg, and L. Wilcox. 2007. On the role of context and prosody in the interpretation and 'okay'. In *ACL*, pages 800–807.

Heylen, D. and R. op den Akker. 2007. Computing backchannel distributions in multi-party conversations. In *ACL:EmbodiedNLP*, pages 17–24.

Johnston, M. 1998. Multimodal language processing. In *ICSLP*.

Jovanovic, N., R. op den Akker, and A. Nijholt. 2006. Adressee identification in face-to-face meetings. In *EACL*.

Jurafsky, D., E. Shriberg, B. Fox, and T. Curl. 1998. Lexical, prosodic and syntactic cures for dialog acts. In *Workshop on Discourse Relations*, pages 114–120.

Kendon, A. 2004. *Gesture: Visible Action as Utterance*. Cambridge University Press.

Kumar, S. and M. Herbert. 2003. Discriminative random fields: A framework for contextual interaction in classification. In *ICCV*.

Lafferty, J., A. McCallum, and F. Pereira. 2001. Conditional random fields: probabilistic models for segmenting and labelling sequence data. In *ICML*.

Maatman, M., J. Gratch, and S. Marsella. 2005. Natural behavior of a listening agent. In *IVA*.

Marcus, Mitchell, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The penn treebank: annotating predicate argument structure. In *ACL:HLT*, pages 114–119.

McNeill, D. 1992. *Hand and Mind: What Gestures Reveal about Thought*. Univ. Chicago Press.

Moore, P.-Y. Hsueh J. 2007. What decisions have you made: Automatic decision detection in conversational speech. In *NAACL-HLT*, pages 25–32.

Morency, Louis-Philippe, Ariadna Quattoni, and Trevor Darrell. 2007. Latent-dynamic discriminative models for continuous gesture recognition. In *CVPR*.

Murray, G. and G. Carenini. 2009. Predicting subjectivity in multimodal conversations. In *EMNLP*, pages 1348–1357.

Nakano, Reinstein, Stocky, and Justine Cassell. 2003. Towards a model of face-to-face grounding. In *ACL*.

Nakano, Y., K. Murata, M. Enomoto, Y. Arimoto, Y. Asa, and H. Sagawa. 2007. Predicting evidence of understanding by monitoring user's task manipulation in multimodal conversations. In *ACL*, pages 121–124.

Nishimura, Ryota, Norihide Kitaoka, and Seiichi Nakagawa. 2007. A spoken dialog system for chat-like conversations considering response timing. *LNCS*, 4629:599–606.

Oviatt, S. 1999. Ten myths of multimodal interaction. *Communications of the ACM*.

Quek, F. 2003. The catchment feature model for multimodal language analysis. In *ICCV*.

Sagae, Kenji and Jun'ichi Tsujii. 2007. Dependency parsing and domain adaptation with LR models and parser ensembles. In *ACL*, pages 1044–1050.

Smith, A., T. Cohn, and M. Osborne. 2005. Logarithmic opinion pools for conditional random fields. In *ACL*, pages 18–25.

Ward, N. and W. Tsukahara. 2000. Prosodic features which cue back-channel responses in english and japanese. *Journal of Pragmatics*, 23:1177–1207.

Text Summarization of Turkish Texts using Latent Semantic Analysis

Makbule Gulcin Ozsoy
Dept. of Computer Eng.
Middle East Tech. Univ.
e1395383@ceng.metu.edu.tr

Ilyas Cicekli
Dept. of Computer Eng.
Bilkent University
ilyas@cs.bilkent.edu.tr

Ferda Nur Alpaslan
Dept. of Computer Eng.
Middle East Tech. Univ.
alpaslan@ceng.metu.edu.tr

Abstract

Text summarization solves the problem of extracting important information from huge amount of text data. There are various methods in the literature that aim to find out well-formed summaries. One of the most commonly used methods is the Latent Semantic Analysis (LSA). In this paper, different LSA based summarization algorithms are explained and two new LSA based summarization algorithms are proposed. The algorithms are evaluated on Turkish documents, and their performances are compared using their ROUGE-L scores. One of our algorithms produces the best scores.

1 Introduction

The exponential growth in text documents brings the problem of finding out whether a text document meets the needs of a user or not. In order to solve this problem, text summarization systems which extract brief information from a given text are created. By just looking at the summary of a document, a user can decide whether the document is of interest to him/her without looking at the whole document.

The aim of a text summarization system is to generate a summary for a given document such that the generated summary contains all necessary information in the text, and it does not include redundant information. Summaries can have different forms (Hahn and Mani, 2000). Extractive summarization systems collect important sentences from the input text in order to generate summaries. Abstractive summarization systems do not collect sentences from the input text, but they try to capture the main concepts in the text, and generate new sentences to represent these main concepts. Abstractive summarization approach is similar to the way that human summarizers follow. Since creating abstractive summaries is a more complex task, most of automatic text summarization systems are extractive summarization systems.

Summarization methods can be categorized according to what they generate and how they generate it (Hovy and Lin, 1999). A summary can be extracted from a single document or from multiple documents. If a summary is generated from a single document, it is known as single-document summarization. On the other hand, if a single summary is generated from multiple documents on the same subject, this is known as multi-document summarization. Summaries are also categorized as generic summaries and query-based summaries. Generic summarization systems generate summaries containing main topics of documents. In query-based summarization, the generated summaries contain the sentences that are related to the given queries.

Extractive summarization systems determine the important sentences of the text in order to put them into the summary. The important sentences of the text are the sentences that represent the main topics of the text. Summarization systems use different approaches to determine the important sentences (Hahn and Mani, 2000; Hovy and Lin, 1999). Some of them look surface clues such as the position of the sentence and the words that are contained in the sentence. Some summarization systems use more semantic oriented analysis such as lexical chains in order to determine the important sentences. Lately, an algebraic method known as Latent Semantic Analysis (LSA) is used in the determination of

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 869–876,
Beijing, August 2010

the important sentences, and successful results are obtained (Gong and Liu, 2001).

In this paper, we present a generic extractive Turkish text summarization system based on LSA. We applied the known text summarization approaches based on LSA in order to extract the summaries of Turkish texts. One of the main contributions of this paper is the introduction of two new summarization methods based on LSA. One of our methods produced much better results than the results of the other known methods.

The rest of the paper is organized as follows. Section 2 presents the related work in summarization. Section 3 explains the LSA approach in detail. Then, the existing algorithms that use different LSA approaches are presented (Gong and Liu, 2001; Steinberger and Jezek 2004; Murray et al., 2005), and two new algorithms are proposed in Section 4. Section 5 presents the evaluation results of these algorithms, and Section 6 presents the concluding remarks.

2 Related Work

Text summarization is an active research area of natural language processing. Its aim is to extract short representative information from input documents. Since the 1950s, various methods are proposed and evaluated. The first studies conducted on text summaries use simple features like terms from keywords/key phrases, terms from user queries, frequency of words, and position of words/sentences (Luhn, 1958).

The use of statistical methods is another approach used for summary extraction. The most well known project that uses statistical approach is the SUMMARIST (Hovy and Lin, 1999). In this project, natural language processing methods are used together with the concept relevance information. The concept relevance information is extracted from dictionaries and WordNet.

Text connectivity is another approach used for summarization. The most well-known algorithm that uses text connectivity is the lexical chains method (Barzilay and Elhadad, 1997; Ercan and Cicekli, 2008). In lexical chains method, WordNet and dictionaries are used to determine semantic relations between words where semantically related words construct lexical chains. Lexical chains are used in the determination of the important sentences of the text.

TextRank (Mihalcea and Tarau, 2004) is a summarization algorithm which is based on graphs, where nodes are sentences and edges represent similarity between sentences. The similarity value is decided by using the overlapping terms. Cluster Lexrank (Qazvinian and Radev, 2008) is another graph-based summarization algorithm, and it tries to find important sentences in a graph in which nodes are sentences and edges are similarities.

In recent years, algebraic methods are used for text summarization. Most well-known algebraic algorithm is Latent Semantic Analysis (LSA) (Landauer et al., 1998). This algorithm finds similarity of sentences and similarity of words using an algebraic method, namely Singular Value Decomposition (SVD). Besides text summarization, the LSA algorithm is also used for document clustering and information filtering.

3 Latent Semantic Analysis

Latent Semantic Analysis (LSA) is an algebraic-statistical method that extracts meaning of words and similarity of sentences using the information about the usage of the words in the context. It keeps information about which words are used in a sentence, while preserving information of common words among sentences. The more common words between sentences mean that those sentences are more semantically related.

LSA method can represent the meaning of words and the meaning of sentences simultaneously. It averages the meaning of words that a sentence contains to find out the meaning of that sentence. It represents the meaning of words by averaging the meaning of sentences that contain this word.

LSA method uses Singular Value Decomposition (SVD) for finding out semantically similar words and sentences. SVD is a method that models relationships among words and sentences. It has the capability of noise reduction, which leads to an improvement in accuracy.

LSA has three main limitations. The first limitation is that it uses only the information in the input text, and it does not use the information of world knowledge. The second limitation is that it does not use the information of word order, syntactic relations, or morphologies. Such information is used for finding out the meaning of words

and texts. The third limitation is that the performance of the algorithm decreases with large and inhomogeneous data. The decrease in performance is observed since SVD which is a very complex algorithm is used for finding out the similarities.

All summarization methods based on LSA use three main steps. These steps are as follows:

1. *Input Matrix Creation:* A matrix which represents the input text is created. The columns of the matrix represent the sentences of the input text and the rows represent the words. The cells are filled out to represent the importance of words in sentences using different approaches, whose details are described in the rest of this section. The created matrix is sparse.
2. *Singular Value Decomposition (SVD):* Singular value decomposition is a mathematical method which models the relationships among terms and sentences. It decomposes the input matrix into three other matrices as follows:

   $A = U \sum V^T$

   where A is the input matrix with dimensions *m x n*, U is an *m x n* matrix which represents the description of the original rows of the input matrix as a vector of extracted concepts, $\sum$ is an *n x n* diagonal matrix containing scaling values sorted in descending order, and V is an *m x n* matrix which represents the description of the original columns of input matrix as a vector of the extracted concepts.
3. *Sentence Selection*: Different algorithms are proposed to select sentences from the input text for summarization using the results of SVD. The details of these algorithms are described in Section 4.

The creation of the input matrix is important for summarization, since it affects the resulting matrices of SVD. There are some ways to reduce the row size of the input matrix, such as eliminating words seen in stop words list, or using the root words only. There are also different approaches to fill out the input matrix cell values, and each of them affects the performance of the summarization system differently. These approaches are as follows:

1. *Number of Occurrence:* The cell is filled with the frequency of the word in the sentence.
2. *Binary Representation of Number of Occurrence:* If the word is seen in the sentence, the cell is filled with 1; otherwise it is filled with 0.
3. *TF-IDF (Term Frequency–Inverse Document Frequency):* The cell is filled with TF-IDF value of the word. This method evaluates the importance of words in a sentence. The importance of a word is high if it is frequent in the sentence, but less frequent in the document. TF-IDF is equal to TF*IDF, and TF and IDF are computed as follows:

   $tf(i,j) = n(i,j) \ / \ \sum_k n(k,j)$

   where $n(i,j)$ is the number of occurrences of the considered word *i* in sentence *j*, and $\sum_k n(k,j)$ is the sum of number of occurrences of all words in sentence *j*.

   $idf(i) = log(|D| / d_i)$

   where $|D|$ is the total number of sentences in the input text, and d_i is the number of sentences where the word *i* appears
4. *Log Entropy*: The cell is filled with log-entropy value of the word, and it is computed as follows.

   $sum = \sum_j p(i,j) \log_2(p(i,j))$

   $global(i) = 1 + (sum / \log_2(n))$

   $local(i,j) = \log_2(1 + f(i,j))$

   $log\text{-}entropy = global * local$

   where $p(i,j)$ is the probability of word *i* that is appeared in sentence *j*, $f(i,j)$ is the number of times word *i* appeared in sentence *j*, and *n* is the number of sentences in the document.
5. *Root Type:* If the root type of the word is noun, the related cell is filled with the frequency of the word in the sentence; otherwise the cell is filled with 0.
6. *Modified TF-IDF:* First the matrix is filled with TF-IDF values. Then, the average TF-IDF values in each row are calculated. If the value in the cell is less than or equal to the average value, the cell value is set to 0. This is our new approach which is proposed to eliminate the noise from the input matrix.

4 Text Summarization

The algorithms in the literature that use LSA for text summarization perform the first two steps of LSA algorithm in the same way. They differ in the way they fill out the input matrix cells.

4.1 Sentence Selection Algorithms in Literature

4.1.1. Gong & Liu (Gong and Liu, 2001)

After performing the first two steps of the LSA algorithm, Gong & Liu summarization algorithm uses V^T matrix for sentence selection. The columns of V^T matrix represent the sentences of the input matrix and the rows of it represent the concepts that are obtained from SVD method. The most important concept in the text is placed in the first row, and the row order indicates the importance of concepts. Cells of this matrix give information about how much the sentence is related to the given concept. A higher cell value means the sentence is more related to the concept.

In Gong & Liu summarization algorithm, the first concept is chosen, and then the sentence most related to this concept is chosen as a part of the resulting summary. Then the second concept is chosen, and the same step is executed. This repetition of choosing a concept and the sentence most related to that concept is continued until a predefined number of sentences are extracted as a part of the summary. In Figure 1, an example V^T matrix is given. First, the concept *con0* is chosen, and then the sentence *sent1* is chosen, since it has the highest cell value in that row.

There are some disadvantages of this algorithm, which are defined by Steinberger and Jezek (2004). First, the reduced dimension size has to be the same as the summary length. This approach may lead to the extraction of sentences from less significant concepts. Second, there exist some sentences that are related to the chosen concept somehow, but do not have the highest cell value in the row of that concept. These kinds of sentences cannot be included in the resulting summary by this algorithm. Third, all chosen concepts are thought to be in the same importance level, but some of those concepts may not be so important in the input text.

	sent0	sent1	sent2	sent3	sent4
con0	0,557	0,691	0,241	0,110	0,432
con1	0,345	0,674	0,742	0,212	0,567
con2	0,732	0,232	0,435	0,157	0,246
con3	0,628	0,836	0,783	0,265	0,343

Figure 1. Gong & Liu approach: From each row of V^T matrix which represents a concept, the sentence with the highest score is selected. This is repeated until a predefined number of sentences are collected.

4.1.2. Steinberger & Jezek (Steinberger and Jezek 2004)

As in the Gong & Liu summarization algorithm, the first two steps of LSA algorithm are executed before selecting sentences to be a part of the resulting summary. For sentence selection, both V and $\sum$ matrixes are used.

The sentence selection step of this algorithm starts with the calculation of the length of each sentence vector which is represented by a row in V matrix. In order to find the length of a sentence vector, only concepts whose indexes are less than or equal to the number of dimension in the new space is used. The dimension of a new space is given as a parameter to the algorithm. The concepts which are highly related to the text are given more importance by using the values in $\sum$ matrix as a multiplication parameter. If the dimension of the new space is *n*, the length of the sentence *i* is calculated as follows:

$$length_i = \sqrt{\sum_{j=1}^{n} V_{ij} * \Sigma_{jj}}$$

After the calculation of sentence lengths, the longest sentences are chosen as a part of the resulting summary. In Figure 2, an example V matrix is given, and the dimension of the new space is assumed to be 3. The lengths of the sentences are calculated using the first three concepts. Since the sentence *sent2* has the highest length, it is extracted first as a part of the summary.

The aim of this algorithm is to get rid of the disadvantages of Gong & Liu summarization algorithm, by choosing sentences which are related to all important concepts and at the same time choosing more than one sentence from an important topic.

	con0	con1	con2	con3	length
sent0	0,846	0,334	0,231	0,210	0,432
sent1	0,455	0,235	0,432	0,342	0,543
sent2	0,562	0,632	0,735	0,857	0,723
sent3	0,378	0,186	0,248	0,545	0,235

Figure 2. Steinberger & Jezek approach: For each row of V matrix, the lengths of sentences using n concepts are calculated. The value n is given as an input parameter. ∑ matrix values are also used as importance parameters in the length calculations.

	sent0	sent1	sent2	sent3	sent4
con0	0,557	0,691	0,241	0,110	0,432
con1	0,345	0,674	0,742	0,212	0,567
con2	0,732	0,232	0,435	0,157	0,246
con3	0,628	0,836	0,783	0,265	0,343

Figure 3. Murray & Renals & Carletta approach: From each row of V^T matrix, concepts, one or more sentences with the higher scores are selected. The number of sentences to be selected is decided by using ∑ matrix.

4.1.3. Murray & Renals & Carletta (Murray et al., 2005)

The first two steps of the LSA algorithm are executed, as in the previous algorithms before the construction of the summary. V^T and ∑ matrices are used for sentence selection.

In this approach, one or more sentences are collected from the topmost concepts in V^T matrix. The number of sentences to be selected depends on the values in the ∑ matrix. The number of sentences to be collected for each topic is determined by getting the percentage of the related singular value over the sum of all singular values, which are represented in the ∑ matrix. In Figure 3, an example V^T matrix is given. Let's choose two sentences from *con0*, and one sentence from *con1*. Thus, the sentences *sent1* and *sent0* are selected from *con0*, and *sent2* is selected from *con1* as a part of the summary.

This approach tries to solve the problems of Gong & Liu's approach. The reduced dimension has not to be same as the number of sentences in the resulting summary. Also, more than one sentence can be chosen even they do not have the highest cell value in the row of the related concept.

4.2 Proposed Sentence Selection Algorithms

The analysis of input documents indicates that some sentences, especially the ones in the introduction and conclusion parts of the documents, belong to more than one main topic. In order to observe whether these sentences are important or they cause noise in matrices of LSA, we propose a new method, named as *Cross*.

Another concern about matrices in LSA is that the concepts that are found after the SVD step may represent main topics or subtopics. So, it is important to determine whether the found concepts are main topics or subtopics. This causes the ambiguity that whether these concepts are subtopics of another main topic, or all the concepts are main topics of the input document. We propose another new method, named as *Topic*, in order to distinguish main topics from subtopics and make sentence selections from main topics.

4.2.1. Cross Method

In this approach, the first two steps of LSA are executed in the same way as the other approaches. As in the Steinberger and Jezek approach, the V^T matrix is used for sentence selection. The proposed approach, however, preprocesses the V^T matrix before selecting the sentences. First, an average sentence score is calculated for each concept which is represented by a row of V^T matrix. If the value of a cell in that row is less than the calculated average score of that row, the score in the cell is set to zero. The main idea is that there can be sentences such that they are not the core sentences representing the topic, but they are related to the topic in some way. The preprocessing step removes the overall effect of such sentences.

After preprocessing, the steps of Steinberger and Jezek approach are followed with a modification. In our Cross approach, first the cell values are multiplied with the values in the ∑ matrix, and the total lengths of sentence vectors, which are represented by the columns of the V^T matrix, are calculated. Then, the longest sentence vectors are collected as a part of the resulting summary.

In Figure 4, an example V^T matrix is given. First, the average scores of all concepts are calculated, and the cells whose values are less than the average value of their row are set to zero.

The boldface numbers are below row averages in Figure 4, and they are set to zero before the calculation of the length scores of sentences. Then, the length score of each sentence is calculated by adding up the concept scores of sentences in the updated matrix. In the end, the sentence *sent1* is chosen for the summary as the first sentence, since it has the highest length score.

	sent0	sent1	sent2	sent3	average
con0	0,557	0,691	**0,241**	**0,110**	0,399
con1	**0,345**	0,674	0,742	**0,212**	0,493
con2	0,732	**0,232**	0,435	**0,157**	0,389
con3	**0,628**	**0,436**	0,783	0,865	0,678
con4	0,557	0,691	**0,241**	0,710	0,549
length	1,846	2,056	1,960	1,575	

Figure 4. Cross approach: For each row of V^T matrix, the cell values are set to zero if they are less than the row average. Then, the cell values are multiplied with the values in the $\sum$ matrix, and the lengths of sentence vectors are found, by summing up all concept values in columns of V^T matrix, which represent the sentences.

4.2.2. Topic Method

The first two steps of LSA algorithm are executed as in the other approaches. For sentence selection, the V^T matrix is used. In the proposed approach, the main idea is to decide whether the concepts that are extracted from the matrix V^T are really main topics of the input text, or they are subtopics. After deciding the main topics which may be a group of subtopics, the sentences are collected as a part of the summary from the main topics.

In the proposed algorithm, a preprocessing step is executed, as in the Cross approach. First, for each concept which is represented by a row of V^T matrix, the average sentence score is calculated and the values less than this score are set to zero. So, a sentence that is not highly related to a concept is removed from the concept in the V^T matrix. Then, the main topics are found. In order to find out the main topics, a *concept x concept* matrix is created by summing up the cell values that are common between the concepts. After this step, the strength values of the concepts are calculated. For this calculation, each concept is thought as a node, and the similarity values in *concept x concept* matrix are considered as edge scores. The strength value of each concept is calculated by summing up the values in each row in *concept x concept* matrix. The topics with the highest strength values are chosen as the main topic of the input text.

	sent0	sent1	sent2	sent3	average
con0	0,557	0,691	**0,241**	**0,110**	0,399
con1	**0,345**	0,674	0,742	**0,212**	0,493
con2	0,732	**0,232**	0,435	**0,157**	0,389
con3	**0,628**	**0,436**	0,783	0,865	0,678
con4	0,557	0,691	**0,241**	0,710	0,549

⇩

	con0	con1	con2	con3	con4	strength
con0	1,248	1,365	1,289	0	2,496	6,398
con1	1,365	1,416	1,177	1,525	1,365	6,848
con2	1,289	1,177	0,732	1,218	1,289	5,705
con3	0	1,525	1,218	1,648	1,575	5,966
con4	2,496	1,365	1,289	1,575	1,958	8,683

⇩

	sent0	sent1	sent2	sent3
con0	0,557	0.691	0	0
con1	0	0,674	0,742	0
con2	0,732	0	0,435	0
con3	0	0	0,783	0,865
con4	0,557	0.691	0	0,710

Figure 5. Topic approach: From each row of V^T matrix, concepts, the values are set to zero if they are less than the row average. Then *concept x concept* similarity matrix is created, and the strength values of concepts are calculated, which show how strong the concepts are related to the other concepts. Then the concept whose strength value is highest is chosen, and the sentence with the highest score from that concept is collected. The sentence selection s repeated until a predefined number of sentences is collected.

After the above steps, sentence selection is performed in a similar manner to Gong and Liu approach. For each main topic selected, the sentence with the highest score is chosen. This selection is done until predefined numbers of sentences are collected.

In Figure 5, an example V^T matrix is given. First, the average scores of each concept is calculated and shown in the last column of the ma-

trix. The cell values that are less than the row average value (boldface numbers in Figure 5) are set to zero. Then, a *concept x concept* matrix is created by filling a cell with the summation of the cell values that are common between those two concepts. The strength values of the concepts are calculated by summing up the concept values, and the strength values are shown in the last column of the related matrix. A higher strength value indicates that the concept is much more related to the other concepts, and it is one of the main topics of the input text. After finding out the main topic which is the concept *con4* in this example, the sentence with the highest cell value which is sentence *sent3* is chosen as a part of the summary.

5 Evaluation

Two different sets of scientific articles in Turkish are used for the evaluation our summarization approach. The articles are chosen from different areas, such as medicine, sociology, psychology, having fifty articles in each set. The second data set has longer articles than the first data set. The abstracts of these articles, which are human-generated summaries, are used for comparison. The sentences in the abstracts may not match with the sentences in the input text. The statistics about these data sets are given in Table 1.

	DS1	DS2
Number of documents	50	50
Sentences per document	89,7	147,3
Words per document	2302,2	3435
Words per sentence	25,6	23,3

Table 1. Statistics of datasets

Evaluation of summaries is an active research area. Judgment of human evaluators is a common approach for the evaluation, but it is very time consuming and may not be objective. Another approach that is used for summarization evaluation is to use the ROUGE evaluation approach (Lin and Hovy, 2003), which is based on n-gram co-occurrence, longest common subsequence and weighted longest common subsequence between the ideal summary and the extracted summary. Although we obtained all ROUGE results (ROUGE-1, ROUGE-2, ROUGE-3, ROUGE-W and ROUGE-L) in our evaluations, we only report ROUGE-L results in this paper. The discussions that are made depending on our ROUGE-L results are also applicable to other ROUGE results. Different LSA approaches are executed using different matrix creation methods.

	G&L	S&J	MRC	Cross	Topic
frequency	0,236	0,250	0,244	0,302	0,244
binary	0,272	0,275	0,274	0,313	0,274
tf-idf	0,200	0,218	0,213	0,304	0,213
logentropy	0,230	0,250	0,235	0,302	0,235
root type	0,283	0,282	0,289	0,320	0,289
mod. tf-idf	0,195	0,221	0,223	0,290	0,223

Table 2. ROUGE-L scores for the data set DS1

In Table 2, it can be observed that the Cross method has the highest ROUGE scores for all matrix creation techniques. The Topic method has the same results with Murray & Renals & Carletta approach, and it is better than the Gong & Liu approach.

Table 2 indicates that all algorithms give their best results when the input matrix is created using the root type of words. Binary and log-entropy approaches also produced good results. Modified tf-idf approach, which is proposed in this paper, did not work well for this data set. The modified tf-idf approach lacks performance because it removes some of the sentences/words from the input matrix, assuming that they cause noise. The documents in the data set DS1 are shorter documents, and most of words/sentences in shorter documents are important and should be kept.

Table 3 indicates that the best F-score is achieved for all when the log-entropy method is used for matrix creation. Modified tf-idf approach is in the third rank for all algorithms. We can also observe that, creating matrix according to the root types of words did not work well for this data set.

Given the evaluation results it can be said that *Cross* method, which is proposed in this paper, is a promising approach. Also *Cross* approach is not affected from the method of matrix creation. It produces good results when it is compared against an abstractive summary which is created by a human summarizer.

	G&L	S&J	MRC	Cross	Topic
frequency	0,256	0,251	0,259	0,264	0,259
binary	0,191	0,220	0,189	0,274	0,189
tf-idf	0,230	0,235	0,227	0,266	0,227
logentropy	0,267	0,245	0,268	0,267	0,268
root type	0,194	0,222	0,197	0,263	0,197
mod. tf-idf	0,234	0,239	0,232	0,268	0,232

Table 3. ROUGE-L scores for the data set DS2

6 Conclusion

The growth of text based resources brings the problem of getting the information matching needs of user. In order to solve this problem, text summarization methods are proposed and evaluated. The research on summarization started with the extraction of simple features and improved to use different methods, such as lexical chains, statistical approaches, graph based approaches, and algebraic solutions. One of the algebraic-statistical approaches is Latent Semantic Analysis method.

In this study, text summarization methods which use Latent Semantic Analysis are explained. Besides well-known Latent Semantic Analysis approaches of Gong & Liu, Steinberger & Jezek and Murray & Renals & Carletta, two new approaches, namely Cross and Topic, are proposed.

Two approaches explained in this paper are evaluated using two different datasets that are in Turkish. The comparison of these approaches is done using the ROUGE-L F-measure score. The results show that the Cross method is better than all other approaches. Another important result of this approach is that it is not affected by different input matrix creation methods.

In future work, the proposed approaches will be improved and evaluated in English texts as well. Also, ideas that are used in other methods, such as graph based approaches, will be used together with the proposed approaches to improve the performance of summarization.

Acknowledgments

This work is partially supported by The Scientific and Technical Council of Turkey Grant "TUBITAK EEEAG-107E151".

References

Barzilay, R. and Elhadad, M. 1997. Using Lexical Chains for Text Summarization. *Proceedings of the ACL/EACL'97 Workshop on Intelligent Scalable Text Summarization*, pages 10-17.

Ercan G. and Cicekli, I. 2008. Lexical Cohesion based Topic Modeling for Summarization. *Proceedings of 9th Int. Conf. Intelligent Text Processing and Computational Linguistics (CICLing-2008)*, pages 582-592.

Gong, Y. and Liu, X. 2001. Generic Text Summarization Using Relevance Measure and Latent Semantic Analysis. *Proceedings of SIGIR'01*.

Hahn, U. and Mani, I. 2000. The challenges of automatic summarization. *Computer*, **33**, 29–36.

Hovy, E. and Lin, C-Y. 1999. Automated Text Summarization in SUMMARIST. I. Mani and M.T. Maybury (eds.), *Advances in Automatic Text Summarization*, The MIT Press, pages 81-94.

Landauer, T.K., Foltz, P.W. and Laham, D. 1998. An Introduction to Latent Semantic Analysis. *Discourse Processes*, 25, 259-284.

Lin, C.Y. and Hovy, E.. 2003. Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics. *Proceedings of 2003 Conf. North American Chapter of the Association for Computational Linguistics on Human Language Technology (HLT-NAACL-2003),* pages 71-78.

Luhn, H.P. 1958. The Automatic Creation of Literature Abstracts. *IBM Journal of Research Development* 2(2), 159-165.

Mihalcea, R. and Tarau, P. 2004. Text-rank - bringing order into texts. *Proceeding of the Conference on Empirical Methods in Natural Language Processing*.

Murray, G., Renals, S. and Carletta, J. 2005. Extractive summarization of meeting recordings. *Proceedings of the 9th European Conference on Speech Communication and Technology*.

Qazvinian, V. and Radev, D.R. 2008. Scientific paper summarization using citation summary networks. *Proceedings of COLING2008*, Manchester, UK, pages 689-696.

Steinberger, J. and Jezek, K. 2004. Using Latent Semantic Analysis in Text Summarization and Summary Evaluation. *Proceedings of ISIM '04*, pages 93-100.

Knowing What to Believe (when you already know something)

Jeff Pasternack **Dan Roth**
University of Illinois, Urbana-Champaign
{jpaster2, danr}@uiuc.edu

Abstract

Although much work in NLP has focused on simply determining what a document *means*, we also must know whether or not to *believe* it. Fact-finding algorithms attempt to identify the "truth" among competing claims in a corpus, but fail to take advantage of the user's prior knowledge and presume that truth itself is universal and objective rather than subjective. We introduce a framework for incorporating prior knowledge into *any* fact-finding algorithm, expressing both general "common-sense" reasoning and specific facts already known to the user as first-order logic and translating this into a tractable linear program. As our results show, this approach scales well to even large problems, both reducing error and allowing the system to determine truth respective to the user rather than the majority. Additionally, we introduce three new fact-finding algorithms capable of outperforming existing fact-finders in many of our experiments.

1 Introduction

Although establishing the trustworthiness of the information presented to us has always been a challenge, the advent of the Information Age and the Internet has made it more critical. Blogs, wikis, message boards and other collaborative media have eliminated the high entry barrier–and, with it, the enforced journalistic standards–of older, established media such as newspapers and television, and even these sometimes loosen their fact-checking in the face of increased competitive pressure. Consequently, we find that corpora derived from these sources now offer far more numerous views of far more questionable veracity. If one author claims Mumbai is the largest city in the world, and another claims it is Seoul, who do we believe? One or both authors could be intentionally lying, honestly mistaken or, alternatively, of different viewpoints of what constitutes a "city" (the city proper? The metropolitan area?) Truth is not objective: there may be many valid definitions of "city", but we should believe the claim that accords with our *user's* viewpoint. Note that the user may be another computational system rather than a human (e.g. building a knowledge base of city sizes for question answering), and often neither the user's nor the information source's perspective will be explicit (e.g. an author will not fully elaborate "the largest city by metropolitan area bounded by...") but will instead be implied (e.g. a user's statement that "I already know the population of city A is X, city B is Y..." implies that his definition of a city accords with these figures).

The most basic approach is to take a vote: if multiple claims are mutually exclusive of each other, select the one asserted by the most sources. In our experiments, sources will be the authors of the document containing the claim, but other sources could be publishers/websites (when no authorship is given), an algorithm that outputs claims, etc. Although sometimes competitive, we found voting to be generally lackluster. A class of algorithms called fact-finders are often a dramatic improvement, but are incapable of taking advantage of the user's prior knowledge. Our framework translates prior knowledge (expressed as first-order logic) into a linear program that constrains the claim beliefs produced by a fact-finder, ensuring that our belief state is consistent with both common sense ("cities usually grow") and known facts ("Los Angeles is more populous than Wichita"). While in the past first-order logic has been translated to NP-hard integer linear programs, we use polynomial-time-solvable linear programs, al-

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 877–885,
Beijing, August 2010

lowing us to readily scale to large problems with extensive prior knowledge, as demonstrated by our experiments.

We next discuss related work, followed by a more in-depth description of the fact-finding algorithms used in our experiments, including three novel, high-performing algorithms: Average·Log, Investment, and PooledInvestment. We then present the framework's mechanics and the translation of first-order logic into a linear program. Finally, we present our experimental setup and results over three domains chosen to illustrate different aspects of the framework, demonstrating that both our new fact-finders and our framework offer performance improvements over the current state of the art.

2 Related Work

The broader field of trust can be split into three areas of interest[1]: theoretical, reputation-based, and information-based.

2.1 Theoretical

Marsh (1994) observes that trust can be global (e.g. eBay's feedback scores), personal (each person has their own trust values), or situational (personal and specific to a context). Fact-finding algorithms are based on global trust, while our framework establishes personal trust by exploiting the user's individual prior knowledge.

Probabilistic logics have been explored as an alternate method of reasoning about trust. Manchala (1998) utilizes fuzzy logic (Novak et al., 1999), an extension of propositional logic permitting [0,1] belief over propositions. Yu and Singh (2003) employs Dempster-Shafer theory (Shafer, 1976), with belief triples (mass, belief, and plausibility) over *sets* of possibilities to permit the modeling of ignorance, while Josang et al. (2006) uses the related subjective logic (Josang, 1997). While our belief in a claim is decidedly Bayesian (the probability that the claim is true), "unknowns" (discussed later) allow us to reason about ignorance as subjective logic and Dempster-Shafer do, but with less complexity.

[1]Following the division proposed by Artz and Gil (2007); see also (Sabater and Sierra, 2005) for a survey from a different perspective.

2.2 Reputation-based

Reputation-based systems determine an entity's trust or standing among peers via transitive recommendations, as PageRank (Brin and Page, 1998) does among web pages, Advogato (Levien, 2008) does among people, and Eigentrust (Kamvar et al., 2003) does among peers in a network. Some, such as Hubs and Authorities (Kleinberg, 1999), are readily adapted to fact-finding, as demonstrated later.

2.3 Information-Based

Information-based approaches utilize content (rather than peer recommendations) to compute trust, and are often specialized for a particular domain. For example, (Zeng et al., 2006) and Wikitrust (Adler and de Alfaro, 2007) determine trust in a wiki's text passages from sequences of revisions but lack the claim-level granularity and general applicability of fact-finders.

Given a large set of sources making conflicting claims, fact-finders determine "the truth" by iteratively updating their parameters, calculating belief in facts based on the trust in their sources, and the trust in sources based on belief in their facts. TruthFinder (Yin et al., 2008) is a straightforward implementation of this idea. AccuVote (Dong et al., 2009a; Dong et al., 2009b) improves on this by using calculated source dependence (where one source derives its information from another) to give higher credibility to independent sources. (Galland et al., 2010)'s 3-Estimates algorithm incorporates the estimated "hardness" of a fact, such that knowing the answer to an easy question earns less trust than to a hard one. Except for AccuVote (whose model of repeated source-to-source copying is inapplicable to our experimental domains) we experimented over all of these algorithms.

3 Fact-Finding

We have a set of sources S each asserting a set of claims C_s, with $C = \bigcup_{s \in S} C_s$. Each claim $c \in C$ belongs to a *mutual exclusion set* $M_c \subseteq C$, a set of claims (including c) that are mutually exclusive with one another; for example, "John was born in 1960" and "John was born in 1965" are mutually exclusive because a person cannot be born in more than one year. If c is not mutually exclusive

to any other claims, then $M_c = \{c\}$. Assuming there exists exactly one true claim $\overline{c}$ in each mutual exclusion set M, our goal is to predict $\overline{c}$ for each M, with accuracy measured by the number of successful predictions divided by the number of mutual exclusion sets, ignoring trivially correct claims that are the sole members of their mutual exclusion set. To this end, fact-finding algorithms iterate to find the trustworthiness of each source $T^i(s)$ at iteration i in terms of the belief in its claims in the previous iteration $B^{i-1}(C_s)$, and belief in each claim $B^i(c)$ in terms of $T^i(S_c)$, where $S_c = \{s : s \in S, c \in C_s\}$ is the set of all sources asserting c. Note that "trustworthiness" and "belief" as used within a fact-finding algorithm typically do not have meaningful semantics (i.e. they are not $[0, 1]$ Bayesian probabilities). Iteration continues until convergence or some predefined stop criteria.

3.1 Priors

Except for 3-Estimates (where the priors are dictated by the algorithm itself), every fact-finder requires priors for $B^0(C)$. For each fact-finder we chose from $B^0_{voted}(c) = |S_c| / \sum_{d \in M_c} |S_d|$, $B^0_{uniform}(c) = 1/|M_c|$, and $B^0_{fixed}(c) = 0.5$.

3.2 Algorithms

3.2.1 Sums (Hubs and Authorities)

Hubs and Authorities (Kleinberg, 1999) gives each page a hub score and an authority score, where its hub score is the sum of the authority of linked pages and its authority is the sum of the hub scores of pages linking to it. This is adapted to fact-finding by viewing sources as hubs (with 0 authority) and claims as authorities (with 0 hub score):

$$T^i(s) = \sum_{c \in C_s} B^{i-1}(c) \quad B^i(c) = \sum_{s \in S_c} T^i(s)$$

We normalize to prevent $T^i(s)$ and $B^i(c)$ from growing unbounded (dividing by $\max_s T^i(s)$ and $\max_c B^i(c)$, respectively), a technique also used with the Investment and Average·Log algorithms (discussed next); this avoids numerical overflow. B^0_{fixed} priors are used.

3.2.2 Average·Log

Computing $T(s)$ as an average of belief in its claims overestimates the trustworthiness of a source with relatively few claims; certainly a source with 90% accuracy over a hundred examples is more trustworthy than a source with 90% accuracy over ten. However, summing the belief in claims allows a source with 10% accuracy to obtain a high trustworthiness score by simply making many claims. Average·Log attempts a compromise, while still using Sums' B^i update rule and B^0_{fixed} priors.

$$T^i(s) = \log |C_s| \cdot \frac{\sum_{c \in C_s} B^{i-1}(c)}{|C_s|}$$

3.2.3 Investment

In the Investment algorithm, sources "invest" their trustworthiness uniformly among their claims. The belief in each claim then grows according to a non-linear function $\mathcal{G}$, and a source's trustworthiness is calculated as the sum of the beliefs in their claims, weighted by the proportion of trust previously contributed to each (relative to the other investors). Since claims with higher-trust sources get higher belief, these claims become relatively more believed and their sources become more trusted. We used $\mathcal{G}(x) = x^g$ with $g = 1.2$ in our experiments, together with B^0_{voted} priors.

$$T^i(s) = \sum_{c \in C_s} B^{i-1}(c) \cdot \frac{T^{i-1}(s)}{|C_s| \cdot \sum_{r \in S_c} \frac{T^{i-1}(r)}{|C_r|}}$$

$$B^i(c) = \mathcal{G}\left(\sum_{s \in S_c} \frac{T^i(s)}{|C_s|} \right)$$

3.2.4 PooledInvestment

Like Investment, sources uniformly invest their trustworthiness in claims and obtain corresponding returns, so $T^i(s)$ remains the same, but now after the belief in the claims of mutual exclusion set M have grown according to $\mathcal{G}$, they are linearly scaled such that the total belief of the claims in M remains the same as it was before applying $\mathcal{G}(x) = x^g$, with $g = 1.4$ and $B^0_{uniform}$ priors used in our experiments. Given $H^i(c) = \sum_{s \in S_c} \frac{T^i(s)}{|C_s|}$, we have:

$$B^i(c) = H^i(c) \cdot \frac{\mathcal{G}(H^i(c))}{\sum_{d \in M_c} \mathcal{G}(H^i(d))}$$

3.3 TruthFinder

TruthFinder (Yin et al., 2008) is pseudoprobabilistic: the basic version of the algorithm below calculates the "probability" of a claim by assuming that each source's trustworthiness is the probability of it being correct and then averages claim beliefs to obtain trustworthiness scores. We also used the "full", more complex TruthFinder, omitted here for brevity. $B^0_{uniform}$ priors are used for both.

$$T^i(s) = \frac{\sum_{c \in C_s} B^{i-1}(c)}{|C_s|}$$
$$B^i(c) = 1 - \prod_{s \in S_c} \left(1 - T^i(s)\right)$$

3.3.1 3-Estimates

3-Estimates (Galland et al., 2010), also omitted for brevity, differs from the other fact-finders by adding a third set of parameters to capture the "difficulty" of a claim, such that correctly asserting a difficult claim confers more trustworthiness than asserting an easy one; knowing the exact population of a city is harder than knowing the population of Mars (presumably 0) and we should not trust a source merely because they provide what is already common knowledge.

4 The Framework

To apply prior knowledge to a fact-finding algorithm, we translate the user's prior knowledge into a linear program. We then iterate the following until convergence or other stopping criteria:

1. Compute $T^i(s)$ for all $s \in S$
2. Compute $B^i(c)$ for all $c \in C$
3. "Correct" beliefs $B^i(C)$ with the LP

4.1 Propositional Linear Programming

To translate prior knowledge into a linear program, we first propositionalize our first-order formulae into propositional logic (Russell and Norvig, 2003). For example, assume we know that Tom is older than John and a person has exactly one age $(\exists_{x,y} Age(Tom, x) \wedge Age(John, y) \wedge x > y) \wedge (\forall_{x,y,z} Age(x, y) \wedge y \neq z \Rightarrow \neg Age(x, z))$, and our system is considering the following claims: $Age(Tom, 30)$, $Age(Tom, 40)$, $Age(John, 25)$, $Age(John, 35)$. Our propositional clauses (after removing redundancies) are then $Age(Tom, 30) \Rightarrow Age(John, 25) \wedge (Age(Tom, 30) \oplus Age(Tom, 40)) \wedge (Age(John, 25) \oplus Age(John, 35))$.

Each claim c will be represented by a proposition, and ultimately a $[0, 1]$ variable in the linear program corresponding, informally, to $P(c)$.[2] Propositionalized constraints have previously been used with *integer* linear programming (ILP) using binary $\{0, 1\}$ values corresponding to $\{false, true\}$, to find an (exact) consistent truth assignment minimizing some cost and solve a global inference problem, e.g. (Roth and Yih, 2004; Roth and Yih, 2007). However, propositional linear programming has two significant advantages:

1. ILP is "winner take all", shifting all belief to one claim in each mutual exclusion set (even when other claims are nearly as plausible) and finding the single most believable consistent *binary assignment*; we instead wish to find a *distribution* of belief over the claims that is consistent with our prior knowledge and as close as possible to the distribution produced by the fact-finder.
2. Linear programs can be solved in polynomial time (e.g. by interior point methods (Karmarkar, 1984)), but ILP is NP-hard.

To create our constraints, we first convert our propositional formula into conjunctive normal form. Then, for each disjunctive clause consisting of a set P of positive literals (claims) and a set N of negations of literals, we add the constraint $\sum_{c \in P} c_v + \sum_{c \in N} (1 - c_v) \geq 1$, where c_v denotes the $[0, 1]$ variable corresponding to each c. The left-hand side is the union bound of at least one of the claims being true (or false, in the case of negated literals); if this bound is at least 1, the constraint is satisfied. This optimism can dilute the strength of our constraints by ignoring potential dependence among claims: $x \Rightarrow y$, $x \vee y$ implies y is true, but since we demand only $y_v \geq x_v$ and $x_v + y_v \geq 1$ we accept any $y_v \geq 0.5$ where

[2] This is a slight mischaracterization, since our linear constraints only *approximate* intersections and unions of events (where each event is "claim c is true"), and we will be satisfying them subject to a linear cost function.

$y_v \geq x_v \geq 1 - y_v$. However, when the claims are mutually exclusive, the union bound is exact; a common constraint is of the form $q \Rightarrow r^1 \vee r^2 \vee \ldots$, where the r literals are mutually exclusive, which translates exactly to $r_v^1 + r_v^2 + \ldots \geq q_v$. Finally, observe that mutual exclusion amongst n claims c^1, c^2, ..., c^n can be compactly written as $c_v^1 + c_v^2 + \ldots + c_v^n = 1$.

4.2 The Cost Function

Having seen how first-order logic can be converted to linear constraints, we now consider the cost function, a distance between the new distribution of belief satisfying our constraints and the original distribution produced by the fact-finder.

First we determine the number of "votes" received by each claim c, computed as $\omega_c = \omega(B(c))$, which should scale linearly with the certainty of the fact-finder's belief in c. Recall that the semantics of the belief score are particular to the fact-finder, so different fact-finders require different vote functions. TruthFinder has pseudo-probabilistic [0,1] beliefs, so we use $\omega_{inv}(x) = \min((1-x)^{-1}, m_{inv})$ with $m_{inv} = 10^{10}$ limiting the maximum number of votes possible; we assume $1/0 = \infty$. ω_{inv} intuitively scales with "error": a belief of 0.99 receives ten times the votes of 0.9 and has a tenth the error (0.01 vs. 0.1). For the remainder of the fact-finders whose beliefs are already "linear", we use the identity function $\omega_{idn}(x) = x$.

The most obvious choice for the cost function might be to minimize "frustrated votes": $\sum_{c \in C} \omega_c(1 - c_v)$. Unfortunately, this results in the linear solver generally assigning 1 to the variable in each mutual exclusion set with the most votes and 0 to all others (except when constraints prevent this), shifting all belief to the highest-vote claim and yielding poor performance. Instead, we wish to satisfy the constraints while keeping each c_v close to ω_c/ω_{M_c}, where $\omega_{M_c} = \sum_{d \in M_c} \omega_d$, and so shift belief among claims as little as possible. We use a weighted Manhattan distance called **VoteDistance**, where the cost for increasing the belief in a claim is proportional to the number of votes against it, and the cost for decreasing belief is proportional to the number of votes for it:

$$\sum_{c \in C} \max \begin{pmatrix} (\omega_{M_c} - \omega_c) \cdot (c_v - \omega_c/\omega_{M_c}), \\ \omega_c \cdot (\omega_c/\omega_{M_c} - c_v) \end{pmatrix}$$

Thus, the belief distribution found by our LP will be the one that satisfies the constraints while simultaneously minimizing the number of votes frustrated by the change from the original distribution. Note that for any linear expressions e and f we can implement $\max(e, f)$ in the objective function by replacing it with a new $[-\infty, \infty]$ helper variable x and adding the linear constraints $x \geq e$ and $x \geq f$.

4.3 From Values to Votes to Belief

Solving the LP gives us $[0, 1]$ values for each variable c_v, but we need to calculate an updated belief $B(c)$. We propose two methods for this:

Vote Conservation: $B(c) = \omega^{-1}(c_v \cdot \omega_{M_c})$
Vote Loss: $B(c) = \omega^{-1}(\min(\omega_c,\ c_v \cdot \omega_{M_c}))$

ω^{-1} is an inverse of the vote function: $\omega_{idn}^{-1}(x) = x$ and $\omega_{inv}^{-1}(x) = 1 - (1+y)^{-1}$. Vote Conservation reallocates votes such that the total number of votes in each mutual exclusion set, ω_M, remains the same after the redistribution. However, if the constraints force c to lose votes, should we believe the other claims in M_c more? Under Vote Loss, a claim can *only* lose votes, ensuring that if other claims in M_c become less believable, c does not itself become more believable relative to claims in other mutual exclusion sets. We found Vote Loss just slightly better on average and used it for all reported results.

4.4 "Unknown" Augmentation

Augmenting our data with "Unknown" claims ensures that every LP is feasible and can be used to model our ignorance given a lack of sufficient information or conflicting constraints. An Unknown claim U_M is added to every mutual exclusion set M (but invisible to the fact-finder) and represents our belief that *none* of the claims in M are sufficiently supported. Now we can write the mutual exclusion constraint for M as $U_M + \sum_{c \in M} c_v = 1$. When propositionalizing FOL, if a disjunctive clause contains a non-negated literal for a claim c, then we add $\vee U_{M_c}$ to the clause.

For example, $Age(John, 35) \Rightarrow Age(Tom, 40)$ becomes $Age(John, 35) \Rightarrow Age(Tom, 40) \vee Age(Tom, Unknown)$. The only exception is when the clause contains claims from only one mutual exclusion set (e.g. "I know Sam is 50 or 60"), and so the LP can only be infeasible if the user directly asserts a contradiction (e.g. "Sam is 50 *and* Sam is 60"). The Unknown itself has a fixed number of votes that cannot be lost; this effectively "smooths" our belief in the claims and imposes a floor for believability. If $Age(Kim, 30)$ has 5 votes, $Age(Kim, 35)$ has 3 votes, and $Age(Kim, Unknown)$ is fixed at 6 votes, we hold that Kim's age is unknown due to lack of evidence. The number of votes that should be given to each Unknown for this purpose depends, of course, on the particular fact-finder and ω function used; in our experiments, we are not concerned with establishing ignorance and thus assign 0 votes.

5 Experiments

Experiments were conducted over three domains (city population, basic biographies, and American vs. British spelling) with four datasets, all using the VoteDistance cost function and Vote Loss vote redistribution. We fixed the number of iterations of the framework (calculating $T^i(S)$, $B^i(S)$ and then solving the LP) at 20, which was found sufficient for all fact-finders. To evaluate accuracy, after the final iteration we look at each mutual exclusion set M and predict the highest-belief claim $c \in M$ (or, if u_M had the highest belief, the second-highest claim), breaking ties randomly, and check that it is the true claim t_M. We omit any M that does not contain a true claim (all known claims are false) and any M that is trivially correct (containing only one claim other than u_M). All results are shown in Table 1. **Vote** is the baseline, choosing either the claim occurring in the most Wikipedia revisions (in the Pop dataset) or claimed by the most sources (for all other datasets). **Sum** is Sums (Hubs and Authorities), **3Est** is 3-Estimates, $\mathbf{TF^s}$ is simplified TruthFinder, $\mathbf{TF^c}$ is "full" TruthFinder, **A·L** is Average·Log, $\mathbf{Inv}^{1.2}$ is Investment with $g = 1.2$, and $\mathbf{Pool}^{1.4}$ is PooledInvestment with $g = 1.4$.

5.1 IBT vs. L+I

We can enforce our prior knowledge against the beliefs produced by the fact-finder in each iteration, or we can apply these constraints just once, after running the fact-finder for 20 iterations without interference. By analogy to (Punyakanok et al., 2005), we refer to these approaches as inference based training (IBT) and learning + inference (L+I), respectively. Our results show that while L+I does better when prior knowledge is not entirely correct (e.g. "Growth" in the city population domain), generally performance is comparable when the effect of the constraints is mild, but IBT can outperform when prior knowledge is vital (as in the spelling domain) by allowing the fact-finder to learn from the provided corrections.

5.2 Wikipedia Infoboxes

To focus on the performance of the framework, we (like previous fact-finding work) naively assume that our data are accurately extracted, but we also require large corpora. Wikipedia Infoboxes (Wu and Weld, 2007) are a semi-structured source covering many domains with readily available authorship, and we produced our city population and basic biographic datasets from the most recent full-history dump of the English Wikipedia (taken January 2008). However, attribution is difficult: if an author edits the page but not the claim within the infobox, is the author implicitly agreeing with (and asserting) the claim? The best performance was achieved by being strict for City Population data, counting only the direct editing of a claim, and lax for Biography data, counting any edit. We hypothesize this is because editors may lack specific knowledge about a city's population (and thus fail to correct an erroneous value) but incorrect birth or death dates are more noticeable.

5.3 Results

5.3.1 City Population

We collected infoboxes for settlements (Geobox, Infobox Settlement, Infobox City, etc.) to obtain 44,761 populations claims qualified by year (e.g. $pop(Denver, 598707, 2008)$), with 4,107 authors total. We took as our "truth" U.S. census data, which gave us 308 non-trivial true facts to test against. Our "common sense" knowledge is that population grows

Table 1: Experimental Results ($\emptyset$ indicates no prior knowledge; all values are percent accuracy) Some results are omitted here (see text). A·L, Inv$^{1.2}$, Pool$^{1.4}$ are our novel algorithms

Dataset	Prior Knowledge	Vote	Sum	3Est	TF^s	TF^c	A·L	Inv$^{1.2}$	Pool$^{1.4}$
Pop	$\emptyset$	81.49	81.82	81.49	82.79	84.42	80.84	**87.99**	80.19
Pop	Growth$_{IBT}$	82.79	79.87	77.92	82.79	**86.36**	80.52	85.39	79.87
Pop	Growth$_{L+I}$	82.79	79.55	77.92	83.44	85.39	80.52	**89.29**	80.84
Pop	Larger$^{2500}_{IBT}$	85.39	85.06	80.52	86.04	87.34	84.74	**89.29**	84.09
Pop	Larger$^{2500}_{L+I}$	85.39	85.06	80.52	86.69	86.69	84.42	**89.94**	84.09
SynPop	$\emptyset$	73.45	87.76	84.87	56.12	87.07	**90.23**	89.41	90.00
SynPop	Pop$\pm 8\%_{IBT}$	88.31	95.46	92.16	**96.42**	95.46	96.15	95.46	**96.42**
SynPop	Pop$\pm 8\%_{L+I}$	88.31	94.77	92.43	82.39	95.32	95.59	**96.29**	96.01
Bio	$\emptyset$	89.80	89.53	89.80	73.04	**90.09**	89.24	88.34	90.01
Bio	CS$_{IBT}$	89.20	89.61	89.20	72.44	89.91	89.35	88.60	**90.20**
Bio	CS$_{L+I}$	89.20	89.61	89.20	57.10	90.09	89.35	88.49	**90.24**
Bio	CS+Decades$_{IBT}$	90.58	90.88	90.58	80.30	91.25	90.91	90.02	**91.32**
Bio	CS+Decades$_{L+I}$	90.58	90.91	90.58	69.27	90.95	90.91	90.09	**91.17**
Spell	$\emptyset$	13.54	9.37	11.96	**41.93**	7.93	10.23	9.36	9.65
Spell	Words$^{100}_{IBT}$	13.69	9.02	12.72	**44.28**	8.05	9.98	11.11	8.86
Spell	Words$^{100}_{L+I}$	13.69	8.86	12.08	**46.54**	8.05	9.98	9.34	7.89
Spell	CS+Words$^{100}_{IBT}$	35.10	31.88	35.10	56.52	29.79	32.85	73.59	**80.68**
Spell	CS+Words$^{100}_{L+I}$	35.10	31.72	34.62	**55.39**	22.06	32.21	30.92	29.95

over time ("Growth" in table 1); therefore, $\forall_{v,w,x,y,z} pop(v,w,y) \wedge pop(v,x,z) \wedge y < z \Rightarrow x > w$. Of course, this often does not hold true: cities can shrink, but performance was nevertheless superior to no prior knowledge whatsoever. The L+I approach does appreciably better because it avoids forcing these sometimes-incorrect constraints onto the claim beliefs while the fact-finder iterates (which would propagate the resulting mistakes), instead applying them only at the end where they can correct more errors than they create. The sparsity of the data plays a role–only a fraction of cities have population claims for multiple years, and those that do are typically larger cities where the correct claim is asserted by an overwhelming majority, greatly limiting the potential benefit of our Growth constraints. We also considered prior knowledge of the relative sizes of some cities, randomly selecting 2500 pairs of them (a, b), where a was more populous than b in year t, asserting $\forall_{x,y} pop(a,x,t) \wedge pop(b,y,t) \Rightarrow x > y$. This "Larger" prior knowledge proved more effective than our oft-mistaken Growth constraint, with modest improvement to the highest-performing Investment fact-finder, and Investment$_{L+I}$ reaches **90.91%** with 10,000 such pairs.

5.3.2 Synthetic City Population

What if attribution were certain and the data more dense? To this end we created a synthetic dataset. We chose 100 random (real) cities and created 100 authors whose individual accuracy a was drawn uniformly from $[0, 1]$. Between 1 and 10 claims (also determined uniformly) were made about each city in each year from 2000 to 2008 by randomly-selected authors. For each city with true population p and year, four incorrect claims were created with populations selected uniformly from $[0.5p, 1.5p]$, each author claiming p with probability a and otherwise asserting one of the four incorrect claims. Our common-sense knowledge was that population did not change by more than 8% per year (also tried on the Wikipedia dataset but with virtually no effect). Like "Growth", "Pop$\pm 8\%$" does not always hold, but a change of more than 8% is much rarer than a shrinking city. These constraints greatly improved results, although we note this would diminish if inaccurate claims had less variance around the true population.

5.3.3 Basic Biographies

We scanned infoboxes to find 129,847 claimed birth dates, 34,201 death dates, 10,418 parent-child pairs, and 9,792 spouses. To get "true" birth and death dates, we extracted data from several online repositories (after satisfying ourselves that they were independent and not derived from Wikipedia!), eliminating any date these sources disagreed upon, and ultimately obtained a total of 2,685 dates to test against. Our common sense ("CS") knowledge was: nobody dies before they are born, people are infertile before the age of 7, nobody lives past 125, all spouses have overlapping lifetimes, no child is born more than a year after a parent's (father's) death, nobody has more than two parents, and nobody is born or dies after 2008 (the "present day", the year of the Wikipedia dump). Applying this knowledge roughly halved convergence times, but had little effect on the results due to data sparsity similar to that seen in the population data–while we know many birthdays and some death dates, relatively few biographies had parent-child and spouse claims. To this we also added knowledge of the decade (but not the exact date) in which 15,145 people were born ("CS+Decades"). Although common sense alone does not notably improve results, it does very well in conjunction with specific knowledge.

5.3.4 American vs. British Spelling

Prior knowledge allows us to find a truth that conforms with the user's viewpoint, even if that viewpoint differs from the norm. After obtaining a list of words with spellings that differed between American and British English (e.g. "color" vs. "colour"), we examined the British National Corpus as well as Washington Post and Reuters news articles, taking the source's (the article author's) use of a disputed word as a claim that his spelling was correct. Our goal was to find the "true" British spellings that conformed to a British viewpoint, but American spellings predominate by far. Consequently, without prior knowledge the fact-finders do very poorly against our test set of 694 British words, predicting American spelling instead in accordance with the great majority of authors (note that accuracy from an American perspective is $1-$"British" accuracy). Next we assumed that the user already knew the correct spelling of 100 random words (removing these from the test set, of course), but with little effect. Finally, we added our common sense ("CS") knowledge: if a spelling a is correct and of length ≥ 4, then if a is a substring of b, $a \Leftrightarrow b$ (e.g. colour $\Leftrightarrow$ colourful). Furthermore, while we do not know a priori whether a spelling is American or British, we do know if e and f are different spellings of the same word, and, if two such spellings have a chain of implication between them, we can break all links in this chain (while some American spellings will still be linked to British spellings, this removes most such errors). Interestingly, common sense alone actually *hurts* results (e.g. PooledInvestment (IBT) gets 6.2%), as it essentially makes the fact-finders more adept at finding the predominant American spellings! However, when some correct spellings are known, results improve greatly and demonstrate IBT's ability to spread strong prior knowledge, easily surpassing L+I. Results improve further with more known spellings (PooledInvestment gets **84.86%** with CS+Words$^{200}_{IBT}$).

6 Conclusion

We have introduced a new framework for incorporating prior knowledge into a fact-finding system, along with several new high-performing fact-finding algorithms (Investment, PooledInvestment, and Average·Log). While the benefits of prior knowledge were most dramatic in the Spelling domain, we saw gains from both "common sense" and specific knowledge in all experiments–even the difficult Biography domain saw faster convergence with common sense alone and notably higher results when specific knowledge was added. We find that while prior knowledge is helpful in reducing error, when the user's viewpoint disagrees with the norm it becomes absolutely essential and, formulated as a linear program, it need not be the computational burden that might otherwise be expected.

Acknowledgements

This research was partly sponsored by the Army Research Laboratory (ARL) (accomplished under Cooperative Agreement Number W911NF-09-2-0053). Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the ARL.

References

Adler, B T and L de Alfaro. 2007. A content-driven reputation system for the Wikipedia. *WWW '07*, 7:261–270.

Artz, D and Y Gil. 2007. A survey of trust in computer science and the Semantic Web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):58–71, June.

Brin, S and L Page. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117.

Dong, X, L Berti-equille, and D Srivastava. 2009a. Integrating conflicting data: the role of source dependence. *Technical report, AT&T Labs-Research, Florham Park, NJ*.

Dong, X.L., L. Berti-Equille, and Divesh Srivastava. 2009b. Truth discovery and copying detection in a dynamic world. *VLDB*, 2(1):562–573.

Galland, Alban, Serge Abiteboul, A. Marian, and Pierre Senellart. 2010. Corroborating information from disagreeing views. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 131–140. ACM.

Josang, A., S. Marsh, and S. Pope. 2006. Exploring different types of trust propagation. *Lecture Notes in Computer Science*, 3986:179.

Josang, A. 1997. Artificial reasoning with subjective logic. *2nd Australian Workshop on Commonsense Reasoning*.

Kamvar, S, M Schlosser, and H Garcia-molina. 2003. The Eigentrust algorithm for reputation management in P2P networks. *WWW '03*.

Karmarkar, N. 1984. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395.

Kleinberg, J M. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632.

Levien, R. 2008. Attack-resistant trust metrics. *Computing with Social Trust*, pages 121–132.

Manchala, D.W. 1998. Trust metrics, models and protocols for electronic commerce transactions. *Proceedings. 18th International Conference on Distributed Computing Systems (Cat. No.98CB36183)*, pages 312–321.

Marsh, S. 1994. Formalising Trust as a Computational Concept. *PhD thesis, University of Stirling*.

Novak, V, I Perfilieva, and J Mockof. 1999. *Mathematical principles of fuzzy logic*. Kluwer Academic Publishers.

Punyakanok, V., D. Roth, W. Yih, and D. Zimak. 2005. Learning and inference over constrained output. In *International Joint Conference on Artificial Intelligence*, volume 19.

Roth, Dan and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL)*, pages 1–8.

Roth, D and W Yih. 2007. Global Inference for Entity and Relation Identification via a Linear Programming Formulation. In Getoor, Lise and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press.

Russell, Stuart and Peter Norvig. 2003. *Artificial Intelligence: A Modern Approach*. Prentice Hall, second edition.

Sabater, Jordi and Carles Sierra. 2005. Review on Computational Trust and Reputation Models. *Artificial Intelligence Review*, 24(1):33–60, September.

Shafer, G. 1976. *A mathematical theory of evidence*. Princeton University Press Princeton, NJ.

Wu, Fei and Daniel S. Weld. 2007. Autonomously semantifying wikipedia. *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management - CIKM '07*, page 41.

Yin, Xiaoxin, Philip S. Yu, and Jiawei Han. 2008. Truth Discovery with Multiple Conflicting Information Providers on the Web. *IEEE Transactions on Knowledge and Data Engineering*, 20(6):796–808.

Yu, Bin and Munindar P. Singh. 2003. Detecting deception in reputation management. *Proceedings of the second international joint conference on Autonomous agents and multiagent systems - AAMAS '03*, page 73.

Zeng, H, M Alhossaini, L Ding, R Fikes, and D L McGuinness. 2006. Computing trust from revision history. *Intl. Conf. on Privacy, Security and Trust*.

Using Web-scale N-grams to Improve Base NP Parsing Performance

Emily Pitler
Computer and Information Science
University of Pennsylvania
epitler@seas.upenn.edu

Shane Bergsma
Department of Computing Science
University of Alberta
sbergsma@ualberta.ca

Dekang Lin
Google, Inc.
lindek@google.com

Kenneth Church
Human Language Technology Center of Excellence
Johns Hopkins University
kenneth.church@jhu.edu

Abstract

We use web-scale N-grams in a base NP parser that correctly analyzes 95.4% of the base NPs in natural text. Web-scale data improves performance. That is, there is no data like more data. Performance scales log-linearly with the number of parameters in the model (the number of unique N-grams). The web-scale N-grams are particularly helpful in harder cases, such as NPs that contain conjunctions.

1 Introduction

Noun phrases (NPs) provide an index to the world's information. About 70% of web queries are NPs (Barr et al., 2008). A robust NP parser could help search engines improve retrieval performance on multi-word NP queries (Zhai, 1997). For example, by knowing the correct parse of "washed (baby carrots)," a search engine could ensure that returned pages (and advertisements) concern clean carrots rather than clean babies. NP structure is also helpful for query expansion and substitution (Jones et al., 2006).

This paper is concerned with base NP parsing. We are given a base NP string as input, and the task is to produce a parse tree as output. Base NPs are NPs that do not contain embedded noun phrases. These are sometimes called NP chunks, or core/non-recursive NPs (Church, 1988; Ramshaw and Marcus, 1995). Correctly parsing (or, equivalently, bracketing) base NPs is challenging because the same part-of-speech (POS) sequence can be parsed differently depending on the specific words involved. For example, "*retired (science teacher)*" and "*(social science) teacher*" have different structures even though they have identical POS sequences.

Lexical statistics are therefore needed in order to parse the above examples, and they must be computed over a lot of text to avoid sparsity. All of our lexical statistics are derived from a new and improved web-scale N-gram corpus (Lin et al., 2010), which we call Google V2.

Despite the importance of base NPs, most sentence parsers do not parse base NPs, since the main training corpus for parsers, the Penn Treebank (PTB) (Marcus et al., 1994), leaves a flat structure for base NPs. Recent annotations by Vadas and Curran (2007a) added NP structure to the PTB. We use these annotations (described in Section 3) for our experiments.

NP parsers usually focus on bracketing three-word noun compounds. Parsing three-word noun compounds is a fairly artificial task; we show that sequences of three nouns make up less than 1% of the three-word-or-longer base NPs in natural text. As the NP length increases, the number of possible binary trees (parses) increases with the Catalan numbers (Church and Patil, 1982). NPs of length three have just two possible parses (chance is 50%), while NPs of length six already have forty-two possible parses (chance is 2%). Long NPs therefore provide much more opportunity to improve performance over the baseline. In Table 1 (Section 7), we show the distribution of base NP length in the PTB. While most NPs are of length three, NP length has a long tail.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 886–894,
Beijing, August 2010

The three-word noun compound assumption also restricts research to the case in which all words are nouns, while base NPs also contain determiners, possessives, adjectives, and conjunctions. Conjunctions and their scopes are particularly challenging. For example, in the NP, "*French television and movie producers*," a parser should conjoin "(*television*) *and* (*movie*)," as opposed to "(*French television*) *and* (*movie*)," "(*French television*) *and* (*movie producers*)" or "(*television*) *and* (*movie producers*)."

To resolve these issues, we train a classifier which uses contextual information from the entire NP and lexical statistics derived from the web-scale N-gram corpus to predict if a given span is a constituent. Our parser then uses this classifier to produce a score for every possible NP-internal bracketing and creates a chart of bracketing scores. This chart can be used as features in a full sentence parser or parsed directly with a chart parser. Our parses are highly accurate, creating a strong new standard for this task.

Finally, we present experiments that investigate the effects of N-gram frequency cutoffs and various sources of N-gram data. We show an interesting relationship between accuracy and the number of unique N-gram types in the data.

2 Related Work

2.1 Three-Word Noun Compounds

The most commonly used data for NP parsing is from Lauer (1995), who extracted 244 three-word noun compounds from the Grolier encyclopedia. When there are only three words, this task reduces to a binary decision:

- Left Branching: * [*retired science*] *teacher*
- Right Branching: *retired* [*science teacher*]

In Lauer (1995)'s set of noun compounds, two-thirds are left branching.

The main approach to these three-word noun compounds has been to compute association statistics between pairs of words and then choose the bracketing that corresponds to the more highly associated pair. The two main models are the *adjacency model* (Marcus, 1980; Liberman and Sproat, 1992; Pustejovsky et al., 1993; Resnik, 1993) and the *dependency model* (Lauer, 1995). Under the adjacency model, the bracketing decision is made by comparing the associations between words one and two versus words two and three (i.e. comparing *retired science* versus *science teacher*). In contrast, the dependency model compares the associations between one and two versus one and three (*retired science* versus *retired teacher*). Lauer (1995) compares the two models and finds the dependency model to be more accurate.

Nakov and Hearst (2005) compute the association scores using frequencies, conditional probabilities, χ^2, and mutual information, for both pairs of words and for linguistically-motivated paraphrases. Lapata and Keller (2005) found that using web-scale data for associations is better than using the (smaller) 100M-word British National Corpus.

2.2 Longer NPs

Focusing on only the three word case misses a large opportunity for base NP parsing. NPs longer than three words commonly occur, making up 29% of our test set. In addition, a chance baseline does exponentially worse as the length of the NP increases. These longer NPs are therefore a major opportunity to improve overall base NP parsing.

Since in the general case, NP parsing can no longer be thought of as a single binary classification problem, different strategies are required.

Barker (1998) reduces the task of parsing longer NPs to making sequential three-word decisions, moving a sliding window along the NP. The window is first moved from right-to-left, inserting right bracketings, and then again from left-to-right, finalizing left bracketings. While Barker (1998) assumes that these three-word decisions can be made in isolation, this is not always valid.[1] Vadas and Curran (2007b) employ Barker's algorithm, but use a supervised classifier to make the sequential bracketing decisions. Because these approaches rely on a sequence of binary decisions,

[1] E.g., although the right-most three words are identical in 1) "*soap opera stars and television producers*," and 2) "*movie and television producers*," the initial right-bracketing decision for "*and television producers*" should be different in each.

early mistakes can cascade and lead to a chain of incorrect bracketings.

Our approach differs from previous work in NP parsing; rather than greedily inserting brackets as in Barker's algorithm, we use dynamic programming to find the global maximum-scoring parse. In addition, unlike previous approaches that have used local features to make local decisions, we use the full NP to score each potential bracketing.

A related line of research aims to *segment* longer phrases that are queried on Internet search engines (Bergsma and Wang, 2007; Guo et al., 2008; Tan and Peng, 2008). Bergsma and Wang (2007) focus on NP queries of length four or greater. They use supervised learning to make segmentation decisions, with features derived from the noun compound bracketing literature. Evaluating the benefits of *parsing* NP queries, rather than simply segmenting them, is a natural application of our system.

3 Annotated Data

Our training and testing data are derived from recent annotations by Vadas and Curran (2007a). The original PTB left a flat structure for base noun phrases. For example, "*retired science teacher*," would be represented as:

(NP (JJ retired) (NN science) (NN teacher))

Vadas and Curran (2007a) annotated NP-internal structure by adding annotations whenever there is a left-bracketing. If no annotations were added, right-branching is assumed. The inter-annotator agreement for exactly matching the brackets on an NP was 98.5%.

This data provides a valuable new resource for parsing research, but little work has so far made use of it. Vadas and Curran (2007b) perform some preliminary experiments on NP bracketing, but use gold standard part-of-speech and named-entity annotations as features in their classifier. Our work establishes a strong and realistic standard on this data; our results will serve as a basis for further research on this topic.

4 Unlabeled N-gram Data

All of our N-gram features described in Section 6.1 rely on probabilities derived from unlabeled data. To use the largest amount of data possible, we exploit web-scale N-gram corpora. N-gram counts are an efficient way to compress large amounts of data (such as all the text on the web) into a manageable size. An N-gram corpus records how often each unique sequence of words occurs. Co-occurrence probabilities can be calculated directly from the N-gram counts. To keep the size manageable, N-grams that occur with a frequency below a particular threshold can be filtered.

The corpus we use is **Google V2** (Lin et al., 2010): a new N-gram corpus with N-grams of length 1-5 that we created from the same 1 trillion word snapshot of the web as Google N-grams Version 1 (Brants and Franz, 2006), but with several enhancements. Duplicate sentences are removed, as well as "sentences" which are probably noise (indicated by having a large proportion of non-alphanumeric characters, being very long, or being very short). Removing duplicate sentences is especially important because automatically-generated websites, boilerplate text, and legal disclaimers skew the source web data, with sentences that may have only been authored once occurring millions of times. We use the suffix array tools described in Lin et al. (2010) to quickly extract N-gram counts.

5 Base NP Parsing Approach

Our goal is to take a base NP string as input and produce a parse tree as output. In practice, it would be most useful if the NP parse could be integrated into a sentence parser. Previous NP parsers are difficult to apply in practice.[2] Work in prepositional phrase attachment that assumes gold-standard knowledge of the competing attachment sites has been criticized as unrealistic (Atterer and Schütze, 2007).

Our system can easily be integrated into full parsers. Its input can be identified quickly and reliably and its output is compatible with downstream parsers.

[2] For example, Vadas and Curran (2007b) report results on NP parsing, but these results include NPs containing prepositional or adverbial phrases (confirmed by personal communication). Practical application of their system would therefore require resolving prepositional phrase attachment as a preprocessing step.

Our parser's input is base NPs, which can be identified with very high accuracy. Kudo and Matsumoto (2001) report 95.8% NP chunking accuracy on PTB data.

Once provided with an NP, our system uses a supervised classifier to predict the probability of a particular contiguous subsequence (span) of the NP being a constituent, given the entire NP as context. This probability can be inserted into the chart that a standard chart parser would use.

For example, the base NP "*French television and movie producers*" would be decomposed into nine different classification problems, scoring the following potential bracketings:

(French television) and movie producers
French (television and) movie producers
(French television and) movie producers ...
French television and (movie producers)

In Section 6, we detail the set of statistical and structural features used by the classifier.

The output of our classifier can be easily used as a feature in a full-sentence structured prediction parser, as in Taskar et al. (2004). Alternatively, our work could be integrated into a full-sentence parser by using our feature representations directly in a discriminative CFG parser (Finkel et al., 2008), or in a parse re-ranker (Ratnaparkhi et al., 1994; Collins and Koo, 2005; Charniak and Johnson, 2005).

While our main objective is to use web-scale lexical statistics to create an accurate classifier for base NP-internal constituents, we do produce a parse tree for evaluation purposes. The probability of a parse tree is defined as the product of the probabilities of all the spans (constituents) in the tree. The most probable tree is computed with the CYK algorithm.

6 Features

Over the course of development experiments, we discovered that the more position-specific our features were, the more effectively we could parse NPs. We define a word's position as its distance from the right of the full NP, as the semantic head of NPs is most often the right-most word. Ultimately, we decided to conjoin each feature with the position of the proposed bracketing. Since the features for differing proposed bracketings are now disjoint, this is equivalent to scoring bracketings with different classifiers, with each classifier chosen according to the bracketing position. We now outline the feature types that are common, but weighted differently, in each proposed bracketing's feature set.

6.1 N-gram Features

All of the features described in this section require estimates of the probability of specific words or sequences of words. All probabilities are computed using **Google V2** (Section 4).

6.1.1 PMI

Recall that the adjacency model for the three-word task uses the associations of the two pairs of adjacent words, while the dependency model uses the associations of the two pairs of attachment sites for the initial noun. We generalize the adjacency and dependency models by including the pointwise mutual information (Church and Hanks, 1990) between *all* pairs of words in the NP:

$$\text{PMI}(x, y) = \log \frac{p(\text{“x y”})}{p(\text{“x”})p(\text{“y”})} \qquad (1)$$

For NPs of length n, for each proposed bracketing, we include separate features for the PMI between all $\binom{n}{2}$ pairs of words in the NP. For NPs including conjunctions, we include additional PMI features (Section 6.1.2).

Since these features are also tied to the proposed bracketing positions (as explained above), this allows us to learn relationships between various associations within the NP and each potential bracketing. For example, consider a proposed bracketing from word 4 to word 5. We learn that a high association of words inside a bracketing (here, a high association between word 4 and word 5) indicates a bracketing is likely, while a high association between words that cross a proposed bracketing (e.g., a high association between word 3 and word 4) indicates the bracketing is unlikely.

The value of these features is the PMI, if it is defined. If the PMI is undefined, we include one of two binary features:
$p(\text{“x y”}) = 0$ or $p(\text{“x”}) \vee p(\text{“y”}) = 0$.

We illustrate the PMI features with an example. In deciding whether (*movie producers*) is a reasonable bracketing within "*French television and movie producers*," the classifier weighs features for all of:

PMI(*French, television*)
PMI(*French, and*)
...
PMI(*television, producers*)
PMI(*and, producers*)
PMI(*movie, producers*)

6.1.2 Conjunctions

Properly handling NPs containing conjunctions (NP+conj) requires special statistical features. For example, *television* and *movie* are commonly conjoined, but the relevant statistics that suggest placing brackets around the phrase "*television and movie*" are not provided by the above PMI features (i.e., this is not clear from PMI(*television, and*), PMI(*television, movie*), nor PMI(*and, movie*)). Rather, we want to know if the full phrase "television and movie" is common.

We thus have additional NP+conj features that consider the PMI association across the word *and*:

$$\text{PMI}_{and}(x, y) = \log \frac{p(\text{“}x \text{ and } y\text{”})}{p(\text{“}x \text{ and”})p(\text{“and } y\text{”})} \quad (2)$$

When PMI_{and} between a pair of words is high, they are likely to be the constituents of a conjunction.

Let *NP*=$(w_1 \dots w_{i-1}, \text{`}and\text{'}, w_{i+1} \dots w_n)$ be an NP+conj. We include the PMI_{and} features between w_{i-1} and all $w \in w_{i+1} \dots w_n$. In the example "*French television and movie producers*," we would include features PMI_{and}(*television, movie*) and PMI_{and}(*television, producers*).

In essence, we are assuming w_{i-1} is the head of one of the items being conjoined, and we score the likelihood of each of the words to the right of the *and* being the head for the other item. In our running example, the conjunction has narrow scope, and PMI_{and}(*television, movie*) is greater than PMI_{and}(*television, producers*), indicating to our classifier that (*television and movie*) is a good bracketing. In other examples the conjunction will join heads that are further apart, as in *((French TV) and (British radio)) stars*, where both of the following hold:

$\text{PMI}_{and}(\text{TV, radio}) > \text{PMI}_{and}(\text{TV, British})$
$\text{PMI}_{and}(\text{TV, radio}) > \text{PMI}_{and}(\text{TV, stars})$

6.2 Lexical

We include a binary feature to indicate the presence of a particular word at each position in the NP. We learn that, for instance, the word *Inc.* in names tends to occur outside of brackets.

6.3 Shape

Previous work on NP bracketing has used gold-standard named entity tags (Vadas and Curran, 2007b) as features. We did not want to use any gold-standard features in our experiments, however NER information is helpful in separating premodifiers from names, i.e. *(news reporter) (Walter Cronkite)*.

As an expedient way to get both NER information and useful information from hyphenated adjectives, abbreviations, and other punctuation, we normalize each string using the following regular expressions:

[A-Z]+ $\rightarrow$ A [a-z]+ $\rightarrow$ a

We use this normalized string as an indicator feature. E.g. the word "Saudi-born" will fire the binary feature "Aa-a."

6.4 Position

We also include the position of the proposed bracketing as a feature. This represents the prior of a particular bracketing, regardless of the actual words.

7 Experiments

7.1 Experimental Details

We use Vadas and Curran (2007a)'s annotations (Section 3) to create training, development and testing data for base NPs, using standard splits of the Penn Treebank (Table 1). We consider all non-trivial base NPs, i.e., those longer than two words.

For training, we expand each NP in our training set into independent examples corresponding to all the possible internal NP-bracketings, and represent these examples as feature vectors (Section 5). Each example is positively labeled if it is

Data Set	Train	Dev	Test	Chance
PTB Section	2-22	24	23	
Length=3	41353	1428	2498	50%
Length=4	12067	445	673	20%
Length=5	3930	148	236	7%
Length=6	1272	34	81	2%
Length>6	616	29	34	$< 1\%$
Total NPs	59238	2084	3522	

Table 1: Breakdown of the PTB base NPs used in our experiments. Chance = 1/Catalan(length).

Features	All NPs	NP+conj	NP-conj
All features	**95.4**	**89.7**	**95.7**
-N-grams	94.0	84.0	94.5
-lexical	92.2	87.4	92.5
-shape	94.9	**89.7**	95.2
-position	95.3	**89.7**	95.6
Right bracketing	72.6	58.3	73.5

Table 2: Accuracy (%) of base NPs parsing; ablation of different feature classes.

consistent with the gold-standard bracketing, otherwise it is a negative example.

We train using LIBLINEAR, an efficient linear Support Vector Machine (SVM).[3] We use an L2-loss function, and optimize the regularization parameter on the development set (reaching an optimum at C=1). We converted the SVM output to probabilities.[4] Perhaps surprisingly, since SVMs are not probabilistic, performance on the development set with these SVM-derived probabilities was higher than using probabilities from the LIBLINEAR logistic regression solver.

At test time, we again expand the NPs and calculate the probability of each constituent, inserting the score into a chart. We run the CYK algorithm to find the most probable parse of the entire NP according to the chart. Our evaluation metric is **Accuracy**: the proportion of times our proposed parse of the NP exactly matches the gold standard.

8 Results

8.1 Base NPs

Our method improves substantially over the baseline of assuming a completely right-branching structure, 95.4% versus 72.6% (Table 2). The accuracy of the constituency classifier itself (before the CYK parser is used) is 96.1%.

The lexical features are most important, but all feature classes are somewhat helpful. In particular, including N-gram PMI features significantly improves the accuracy, from 94.0% to 95.4%.[5] Correctly parsing more than 19 base NPs out of 20 is an exceptional level of accuracy, and provides a strong new standard on this task. The most comparable result is by Vadas and Curran (2007b), who achieved 93.0% accuracy on a different set of PTB noun phrases (see footnote 2), but their classifier used features based on gold-standard part-of-speech and named-entity information.

Exact match is a tough metric for parsing, and the difficulty increases as the length of the NP increases (because there are more decisions to make correctly). At three word NPs, our accuracy is 98.5%; by six word NPs, our accuracy drops to 79.0% (Figure 1). Our method's accuracy decreases as the length of the NP increases, but much less rapidly than a right-bracketing or chance baseline.

8.2 Base NPs with Conjunctions

N-gram PMI features help more on NP+conj than on those that do not contain conjunctions (NP-conj) (Table 2). N-gram PMI features are the most important features for NP+conj, increasing accuracy from 84.0% to 89.7%, a 36% relative reduction in error.

8.3 Effect of Thresholding N-gram data

We now address two important related questions: 1) how does our parser perform as the amount of unlabeled auxiliary data varies, and 2) what is the effect of thresholding an N-gram corpus? The second question is of widespread relevance as

[3] `www.csie.ntu.edu.tw/~cjlin/liblinear/`

[4] Following instructions in `http://www.csie.ntu.edu.tw/~cjlin/liblinear/FAQ.html`

[5] McNemar's test, $p < 0.05$

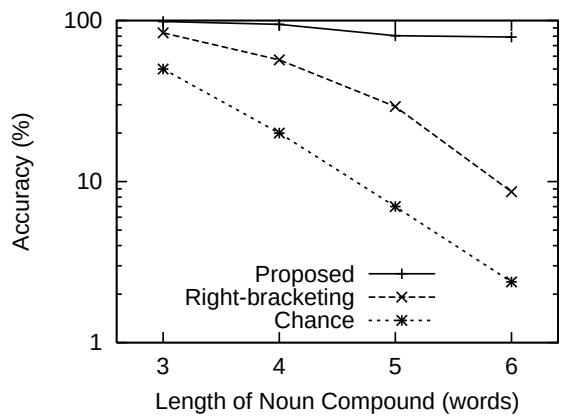


Figure 1: Accuracy (log scale) over different NP lengths, of our method, the right-bracketing baseline, and chance (1/Catalan(length)).

thresholded N-gram corpora are now widely used in NLP. Without thresholds, web-scale N-gram data can be unmanageable.

While we cannot lower the threshold after creating the N-gram corpus, we can raise it, filtering more N-grams, and then measure the relationship between threshold and performance.

Threshold	Unique N-grams	Accuracy
10	4,145,972,000	95.4%
100	391,344,991	95.3%
1,000	39,368,488	95.2%
10,000	3,924,478	94.8%
100,000	386,639	94.8%
1,000,000	37,567	94.4%
10,000,000	3,317	94.0%

Table 3: There is no data like more data. Accuracy improves with the number of parameters (unique N-grams).

We repeat the parsing experiments while including in our PMI features only N-grams with a count $\geq$10 (the whole data set), $\geq$100, $\geq$1000, ..., $\geq 10^7$. All other features (lexical, shape, position) remain unchanged. The N-gram data almost perfectly exhibits Zipf's power law: raising the threshold by a factor of ten decreases the number of unique N-grams by a factor of ten (Table 3). The improvement in accuracy scales log-linearly with the number of unique N-grams. From a practical standpoint, we see a trade-off between storage and accuracy. There are consistent improvements in accuracy from lowering the threshold and increasing the amount of auxiliary data. If for some application it is necessary to reduce storage by several orders of magnitude, then one can easily estimate the resulting impact on performance.

Corpus	# of tokens	τ	# of types
NEWS	3.2 B	1	3.7 B
Google V1	1,024.9 B	40	3.4 B
Google V2	207.4 B	10	4.1 B

Table 4: N-gram data, with total number of words (*tokens*) in the original corpus (in billions, B), frequency threshold used to filter the data, τ, and total number of unique N-grams (*types*) remaining in the data after thresholding.

We repeat the thresholding experiments using two other N-gram sources:

NEWS: N-gram data created from a large set of news articles including the Reuters and Gigaword (Graff, 2003) corpora, not thresholded.

Google V1: The original web-scale N-gram corpus (Section 4).

Details of these sources are given in Table 4.

For a given number of unique N-grams, using any of the three sources does about the same (Figure 2). It does not matter that the source corpus for Google V1 is about five times larger than the source corpus for Google V2, which in turn is sixty-five times larger than NEWS (Table 4). Accuracies increase linearly with the log of the number of *types* in the auxiliary data set.

Google V1 is the one data source for which the relationship between accuracy and number of N-grams is not monotonic. After about 100 million unique N-grams, performance starts decreasing. This drop shows the need for Google V2. Since Google V1 contains duplicated web pages and sentences, mistakes that should be rare can appear to be quite frequent. Google V2, which comes from the same snapshot of the web as Google V1, but has only unique sentences, does not show this drop.

We regard the results in Figure 2 as a companion to Banko and Brill (2001)'s work on exponentially increasing the amount of labeled training data. Here we see that varying the amount of

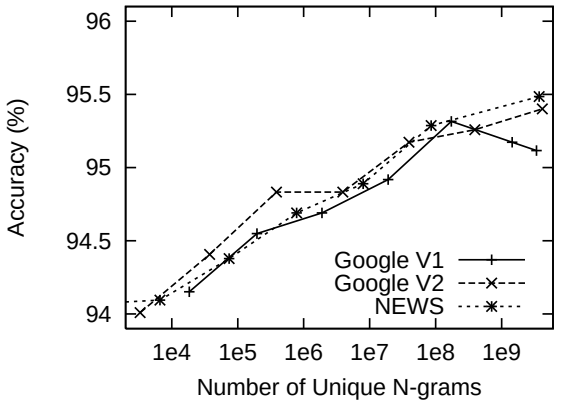


Figure 2: There is no data like more data. Accuracy improves with the number of parameters (unique N-grams). This trend holds across three different sources of N-grams.

unlabeled data can cause an equally predictable improvement in classification performance, without the cost of labeling data.

Suzuki and Isozaki (2008) also found a log-linear relationship between unlabeled data (up to a billion words) and performance on three NLP tasks. We have shown that this trend continues well beyond Gigaword-sized corpora. Brants et al. (2007) also found that more unlabeled data (in the form of input to a language model) leads to improvements in BLEU scores for machine translation.

Adding noun phrase parsing to the list of problems for which there is a "bigger is better" relationship between performance and unlabeled data shows the wide applicability of this principle. As both the amount of text on the web and the power of computer architecture continue to grow exponentially, collecting and exploiting web-scale auxiliary data in the form of N-gram corpora should allow us to achieve gains in performance linear in time, without any human annotation, research, or engineering effort.

9 Conclusion

We used web-scale N-grams to produce a new standard in performance of base NP parsing: 95.4%. The web-scale N-grams substantially improve performance, particularly in long NPs that include conjunctions. There is no data like more data. Performance improves log-linearly with the number of parameters (unique N-grams). One can increase performance with larger models, e.g., increasing the size of the unlabeled corpora, or by decreasing the frequency threshold. Alternatively, one can decrease storage costs with smaller models, e.g., decreasing the size of the unlabeled corpora, or by increasing the frequency threshold. Either way, the log-linear relationship between accuracy and model size makes it easy to estimate the trade-off between performance and storage costs.

Acknowledgments

We gratefully acknowledge the Center for Language and Speech Processing at Johns Hopkins University for hosting the workshop at which this research was conducted.

References

Atterer, M. and H. Schütze. 2007. Prepositional phrase attachment without oracles. *Computational Linguistics*, 33(4):469–476.

Banko, M. and E. Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In *ACL*.

Barker, K. 1998. A trainable bracketer for noun modifiers. In *Twelfth Canadian Conference on Artificial Intelligence (LNAI 1418)*.

Barr, C., R. Jones, and M. Regelson. 2008. The linguistic structure of English web-search queries. In *EMNLP*.

Bergsma, S. and Q.I. Wang. 2007. Learning noun phrase query segmentation. In *EMNLP-CoNLL*.

Brants, T. and A. Franz. 2006. The Google Web 1T 5-gram Corpus Version 1.1. LDC2006T13.

Brants, T., A.C. Popat, P. Xu, F.J. Och, and J. Dean. 2007. Large language models in machine translation. In *EMNLP*.

Charniak, E. and M. Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *ACL*.

Church, K.W. and P. Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.

Church, K. and R. Patil. 1982. Coping with syntactic ambiguity or how to put the block in the box on the table. *Computational Linguistics*, 8(3-4):139–149.

Church, K.W. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *ANLP*.

Collins, M. and T. Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70.

Finkel, J.R., A. Kleeman, and C.D. Manning. 2008. Efficient, feature-based, conditional random field parsing. In *ACL*.

Graff, D. 2003. English Gigaword. LDC2003T05.

Guo, J., G. Xu, H. Li, and X. Cheng. 2008. A unified and discriminative model for query refinement. In *SIGIR*.

Jones, R., B. Rey, O. Madani, and W. Greiner. 2006. Generating query substitutions. In *WWW*.

Kudo, T. and Y. Matsumoto. 2001. Chunking with support vector machines. In *NAACL*.

Lapata, M. and F. Keller. 2005. Web-based models for natural language processing. *ACM Transactions on Speech and Language Processing*, 2(1):1–31.

Lauer, M. 1995. Corpus statistics meet the noun compound: some empirical results. In *ACL*.

Liberman, M. and R. Sproat. 1992. The stress and structure of modified noun phrases in English. *Lexical matters*, pages 131–181.

Lin, D., K. Church, H. Ji, S. Sekine, D. Yarowsky, S. Bergsma, K. Patil, E. Pitler, R. Lathbury, V. Rao, K. Dalwani, and S. Narsale. 2010. New tools for web-scale n-grams. In *LREC*.

Marcus, M.P., B. Santorini, and M.A. Marcinkiewicz. 1994. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Marcus, M.P. 1980. *Theory of Syntactic Recognition for Natural Languages*. MIT Press, Cambridge, MA, USA.

Nakov, P. and M. Hearst. 2005. Search engine statistics beyond the n-gram: Application to noun compound bracketing. In *CoNLL*.

Pustejovsky, J., P. Anick, and S. Bergler. 1993. Lexical semantic techniques for corpus analysis. *Computational Linguistics*, 19(2):331–358.

Ramshaw, L.A. and M.P. Marcus. 1995. Text chunking using transformation-based learning. In *3rd ACL Workshop on Very Large Corpora*.

Ratnaparkhi, A., S. Roukos, and R.T. Ward. 1994. A maximum entropy model for parsing. In *Third International Conference on Spoken Language Processing*.

Resnik, P. 1993. *Selection and information: a class-based approach to lexical relationships*. Ph.D. thesis, University of Pennsylvania.

Suzuki, J. and H. Isozaki. 2008. Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. In *ACL*.

Tan, B. and F. Peng. 2008. Unsupervised query segmentation using generative language models and Wikipedia. In *WWW*.

Taskar, B., D. Klein, M. Collins, D. Koller, and C. Manning. 2004. Max-margin parsing. In *EMNLP*.

Vadas, D. and J.R. Curran. 2007a. Adding noun phrase structure to the Penn Treebank. In *ACL*.

Vadas, D. and J.R. Curran. 2007b. Large-scale supervised models for noun phrase bracketing. In *PACLING*.

Zhai, C. 1997. Fast statistical parsing of noun phrases for document indexing. In *ANLP*.

Citation Summarization Through Keyphrase Extraction

Vahed Qazvinian
Department of EECS
University of Michigan
vahed@umich.edu

Dragomir R. Radev
School of Information and
Department of EECS
University of Michigan
radev@umich.edu

Arzucan Özgür
Department of EECS
University of Michigan
ozgur@umich.edu

Abstract

This paper presents an approach to summarize single scientific papers, by extracting its contributions from the set of citation sentences written in other papers. Our methodology is based on extracting significant keyphrases from the set of citation sentences and using these keyphrases to build the summary. Comparisons show how this methodology excels at the task of single paper summarization, and how it out-performs other multi-document summarization methods.

1 Introduction

In recent years statistical physicists and computer scientists have shown great interest in analyzing complex adaptive systems. The study of such systems can provide valuable insight on the behavioral aspects of the involved agents with potential applications in economics and science. One such aspect is to understand what motivates people to provide the $n+1^{st}$ review of an artifact given that they are unlikely to add something significant that has not already been said or emphasized. Citations are part of such complex systems where articles use citations as a way to mention different contributions of other papers, resulting in a collective system.

The focus of this work is on the corpora created based on citation sentences. A citation sentence is a sentence in an article containing a citation and can contain zero or more *nuggets* (i.e., non-overlapping contributions) about the cited article. For example the following sentences are a few citation sentences that appeared in the NLP literature in past that talk about Resnik's work.

The STRAND system **(Resnik, 1999)***, for example, uses structural markup information from the pages, without looking at their content, to attempt to align them.*

Resnik (1999) *addressed the issue of language identification for finding Web pages in the languages of interest.*

Mining the Web for bilingual text **(Resnik, 1999)** *is not likely to provide sufficient quantities of high quality data..*

The set of citations is important to analyze because human summarizers have put their effort collectively but independently to read the target article and cite its important contributions. This has been shown in other work too (Elkiss et al., 2008; Nanba et al., 2004; Qazvinian and Radev, 2008; Mei and Zhai, 2008; Mohammad et al., 2009). In this work, we introduce a technique to summarize the set of citation sentences and cover the major contributions of the target paper. Our methodology first finds the set of keyphrases that represent important information units (i.e., nuggets), and then finds the best set of k sentences to cover more, and more important nuggets.

Our results confirm the effectiveness of the method and show that it outperforms other state of the art summarization techniques. Moreover, as shown in the paper, this method does not need to calculate the full cosine similarity matrix for a document cluster, which is the most time consuming part of the mentioned baseline methods.

1.1 Related Work

Previous work has used citations to produce summaries of scientific work (Qazvinian and Radev,

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 895–903,
Beijing, August 2010

2008; Mei and Zhai, 2008; Elkiss et al., 2008). Other work (Bradshaw, 2003; Bradshaw, 2002) benefits from citations to determine the content of articles and introduce "Reference Directed Indexing" to improve the results of a search engine.

In other work, (Nanba and Okumura, 1999) analyze citation sentences and automatically categorize citations into three groups using 160 predefined phrase-based rules to support a system for writing a survey. Previous research has shown the importance of the citation summaries in understanding what a paper contributes. In particular, (Elkiss et al., 2008) performed a large-scale study on citation summaries and their importance. Results from this experiment confirmed that the "Self Cohesion" (Elkiss et al., 2008) of a citation summary of an article is consistently higher than the that of its abstract and that citations contain additional information that does not appear in abstracts.

Kan et al. (2002) use annotated bibliographies to cover certain aspects of summarization and suggest using metadata and critical document features as well as the prominent content-based features to summarize documents. Kupiec et al. (1995) use a statistical method and show how extracts can be used to create summaries but use no annotated metadata in summarization.

Siddharthan and Teufel describe a new task to decide the scientific attribution of an article (Siddharthan and Teufel, 2007) and show high human agreement as well as an improvement in the performance of Argumentative Zoning (Teufel, 2005). Argumentative Zoning is a rhetorical classification task, in which sentences are labeled as one of Own, Other, Background, Textual, Aim, Basis, Contrast according to their role in the author's argument. These all show the importance of citation summaries and the vast area for new work to analyze them to produce a summary for a given topic.

The Maximal Marginal Relevance (MMR) summarization method, which is based on a greedy algorithm, is described in (Carbonell and Goldstein, 1998). MMR uses the full similarity matrix to choose the sentences that are the least similar to the sentences already selected for the summary. We selected this method as one of our baseline methods, which we have explained in more details in Section 4.

Fact	Occurrences
f_1: " Supervised Learning"	5
f_2: " instance/concept relations"	3
f_3: "Part-of-Speech tagging"	3
f_4: "filtering QA results"	2
f_5: "lexico-semantic information"	2
f_6: "hyponym relations"	2

Table 2: Nuggets of P03-1001 extracted by annotators.

2 Data

In order to evaluate our method, we use the ACL Anthology Network (AAN), which is a collection of papers from the Computational Linguistics journal and proceedings from ACL conferences and workshops and includes more than $13,000$ papers (Radev et al., 2009). We use 25 manually annotated papers from (Qazvinian and Radev, 2008), which are highly cited articles in AAN. Table 1 shows the ACL ID, title, and the number of citation sentences for these papers.

The annotation guidelines asked a number of annotators to read the citation summary of each paper and extract a list of the main contributions of that paper. Each item on the list is a non-overlapping contribution (nugget) perceived by reading the citation summary. The annotation strictly instructed the annotators to focus on the citing sentences to do the task and not their own background on the topic. Then, extracted nuggets are reviewed and those nuggets that have only been mentioned by 1 annotator are removed. Finally, the union of the rest is used as a set of nuggets representing each paper.

Table 2 lists the nuggets extracted by annotators for P03-1001.

3 Methodology

Our methodology assumes that each citation sentence covers 0 or more nuggets about the cited papers, and tries to pick sentences that maximize nugget coverage with respect to summary length.

These nuggets are essentially represented using keyphrases. Therefore, we try to extract significant keyphrases in order to represent nuggets each sentence contains. Here, the keyphrases are ex-

ACL-ID	Title	# citations
N03-1017	Statistical Phrase-Based Translation	180
P02-1006	Learning Surface Text Patterns For A Question Answering System	74
P05-1012	On-line Large-Margin Training Of Dependency Parsers	71
C96-1058	Three New Probabilistic Models For Dependency Parsing: An Exploration	66
P05-1033	A Hierarchical Phrase-Based Model For Statistical Machine Translation	65
P97-1003	Three Generative, Lexicalized Models For Statistical Parsing	55
P99-1065	A Statistical Parser For Czech	54
J04-4002	The Alignment Template Approach To Statistical Machine Translation	50
D03-1017	Towards Answering Opinion Questions: Separating Facts From Opinions ...	42
P05-1013	Pseudo-Projective Dependency Parsing	40
W00-0403	Centroid-Based Summarization Of Multiple Documents: Sentence Extraction, ...	31
P03-1001	Offline Strategies For Online Question Answering: Answering Questions Before They Are Asked	27
N04-1033	Improvements In Phrase-Based Statistical Machine Translation	24
A00-2024	Cut And Paste Based Text Summarization	20
W00-0603	A Rule-Based Question Answering System For Reading Comprehension Tests	19
A00-1043	Sentence Reduction For Automatic Text Summarization	19
C00-1072	The Automated Acquisition Of Topic Signatures For Text Summarization	19
W05-1203	Measuring The Semantic Similarity Of Texts	17
W03-0510	The Potential And Limitations Of Automatic Sentence Extraction For Summarization	15
W03-0301	An Evaluation Exercise For Word Alignment	14
A00-1023	A Question Answering System Supported By Information Extraction	13
D04-9907	Scaling Web-Based Acquisition Of Entailment Relations	12
P05-1014	The Distributional Inclusion Hypotheses And Lexical Entailment	10
H05-1047	A Semantic Approach To Recognizing Textual Entailment	8
H05-1079	Recognising Textual Entailment With Logical Inference	9

Table 1: List of papers chosen from AAN for evaluation together with the number of sentences citing each.

	unique	all	max freq
unigrams	229,631	7,746,792	437,308
bigrams	2,256,385	7,746,791	73,957
3-grams	5,125,249	7,746,790	3,600
4-grams	6,713,568	7,746,789	2,408

Table 3: Statistics on the abstract corpus in AAN used as the background data

pressed using N-grams, and thus these building units are the key elements to our summarization. For each citation sentence d_i, our method first extracts a set of important keyphrases, D_i, and then tries to find sentences that have a larger number of important and non-redundant keyphrases. In order to take the first step, we extract statistically significantly frequent N-grams (up to $N = 4$) from each citing sentence and use them as the set of representative keyphrases for that citing sentence.

3.1 Automatic Keyphrase Extraction

A list of keyphrases for each citation sentence can be generated by extracting N-grams that occur significantly frequently in that sentence compared to a large corpus of such N-grams. Our method for such an extraction is inspired by the previous work by Tomokiyo and Hurst (Tomokiyo and Hurst, 2003).

A language model, $\mathcal{M}$, is a statistical model that assigns probabilities to a sequence of N-grams. Every language model is a probability distribution over all N-grams and thus the probabilities of all N-grams of the same length sum up to 1. In order to extract keyphrases from a text using statistical significance we need two language models. The first model is referred to as the *Background Model ($\mathcal{BM}$)* and is built using a large text corpus. Here we build the BM using the text of all the paper abstracts provided in AAN[1]. The second language model is called the *Foreground Model ($\mathcal{FM}$)* and is the model built on the text from which keyphrases are being extracted. In this work, the set of all citation sentences that cite a particular target paper are used to build a foreground language model.

Let g^i be an N-gram of size i and $C_{\mathcal{M}}(g^i)$ denote the count of g^i in the model $\mathcal{M}$. First, we extract the counts of each N-grams in both the background ($\mathcal{BM}$) and the foreground corpora ($\mathcal{FM}$).

[1]http://chernobog.si.umich.edu/clair/anthology/index.cgi

$$
\begin{aligned}
M_{\mathcal{BM}} &= \sum_{g^i \in \{\mathcal{BM} \cup \mathcal{FM}\}} 1 \\
N_{\mathcal{BM}} &= \sum_{g^i \in \{\mathcal{BM} \cup \mathcal{FM}\}} C_{\mathcal{BM}}(g^i) \\
N_{\mathcal{FM}} &= \sum_{g^i \in \mathcal{FM}} C_{\mathcal{FM}}(g^i) \\
\hat{p}_{\mathcal{FM}}(g^i) &= C_{\mathcal{FM}}(g^i)/N_{\mathcal{FM}} \\
\hat{p}_{\mathcal{BM}}(g^i) &= (C_{\mathcal{BM}}(g^i)+1)/(M_{\mathcal{BM}}+N_{\mathcal{BM}})
\end{aligned}
$$

The last equation is also known as Laplace smoothing (Manning and Schutze, 2002) and handles the N-grams in the foreground corpus that have a 0 occurrence frequency in the background corpus. Next, we extract N-grams from the foreground corpus that have significant frequencies compared to the frequency of the same N-grams in the background model and its individual terms in the foreground model.

To measure how randomly a set of consecutive terms are forming an N-gram, Tomokiyo and Hurst (Tomokiyo and Hurst, 2003) use pointwise divergence. In particular, for an N-gram of size i, $g^i = (w_1 w_2 \cdots w_i)$,

$$
\delta_{g^i}(\mathcal{FM}^i \| \mathcal{FM}^1) = \hat{p}_{\mathcal{FM}}(g^i) \log\left(\frac{\hat{p}_{\mathcal{FM}}(g^i)}{\prod_{j=1}^{i} \hat{p}_{\mathcal{FM}}(w_j)}\right)
$$

This equation shows the extent to which the terms forming g^i have occurred together randomly. In other words, it indicates the extent of information that we lose by assuming independence of each word by applying the unigram model, instead of the N-gram model.

In addition, to measure how randomly a sequence of words appear in the foreground model with respect to the background model, we use pointwise divergence as well. Here, pointwise divergence defines how much information we lose by assuming that g^i is drawn from the background model instead of the foreground model:

$$
\delta_{g^i}(\mathcal{FM}^i \| \mathcal{BM}^i) = \hat{p}_{\mathcal{FM}}(g^i) \log\left(\frac{\hat{p}_{\mathcal{FM}}(g^i)}{\hat{p}_{\mathcal{BM}}(g^i)}\right)
$$

(Corley and Mihalcea, 2005) applied or utilized lexical based word overlap measures.
{overlap measures, word overlap, lexical based, utilized lexical}

Table 4: Example: citation sentence for W05-1203 written by D06-1621, and its extracted bigrams.

We set the criteria of choosing a sequence of words as significant to be whether it has positive pointwise divergence with respect to both the background model, and individual terms of the foreground model. In other words we extract all g^i from $\mathcal{FM}$ for which the both properties are positive:

$$
\begin{aligned}
\delta_{g^i}(\mathcal{FM}^i \| \mathcal{BM}^i) &> 0 \\
\delta_{g^i}(\mathcal{FM}^i \| \mathcal{FM}^1) &\geq 0
\end{aligned}
$$

The equality condition in the second equation is specifically set to handle unigrams, in which $\hat{p}_{\mathcal{FM}}(g^i) = \prod_{j=1}^{i} \hat{p}_{\mathcal{FM}}(w_j)$.

In order to handle the text corpora and building the language models, we have used the CMU-Cambridge Language Model toolkit (Clarkson and Rosenfeld, 1997). We use the set of citation sentences for each paper to build foreground language models. Furthermore, we employ this tool and make the background model using nearly 11,000 abstracts from AAN. Table 3 summarizes some of the statistics about the background data.

Once keyphrases (significant N-grams) of each sentence are extracted, we remove all N-grams in which more than half of the terms are stopwords. For instance, we remove all stopword unigrams, if any, and all bigrams with at least one stopword in them. For 3-grams and 4-grams we use a threshold of 2 and 3 stopwords respectively. After that, the set of remaining N-grams is used to represent each sentence and to build summaries. Table 4 shows an example of a citation sentence from D06-1621 citing W05-1203 (Corley and Mihalcea, 2005), and its extracted bigrams.

3.2 Sentence Selection

After extracting the set of keyphrases for each sentence, d_i, the sentence is represented using its set

of N-grams, denoted by D_i. Then, the goal is to pick sentences (sets) for each paper that cover more important and non-redundant keyphrases. Essentially, keyphrases that have been repeated in more sentences are more important and could represent more important nuggets. Therefore, sentences that contain more frequent keyphrases are more important. Based on this intuition we define the reward of building a summary comprising a set of keyphrases S as

$$f(S) = |S \cap A|$$

where A is the set of all keyphrases from sentences not in the summary.

The set function f has three main properties. First, it is non-negative. Second, it is monotone (i.e., For every set v we have $f(S + v) \geq f(S)$). Third, f is sub-modular. The submodularity means that for a set v and two sets $S \subseteq T$ we have

$$f(S + v) - f(S) \geq f(T + v) - f(T)$$

Intuitively, this property implies that adding a set v to S will increase the reward at least as much as it would to a larger set T. In the summarization setting, this means that adding a sentence to a smaller summary will increase the reward of the summary at least as much as adding it to a larger summary that subsumes it. The following theorem formalizes this and is followed by a proof.

Theorem 1 *The reward function f is submodular.*

Proof
We start by defining a gain function $\mathcal{G}$ of adding sentence (set) D_i to $\mathcal{S}_{k-1}$ where $\mathcal{S}_{k-1}$ is the set of keyphrases in a summary built using $k-1$ sentences, and D_i is a candidate sentence to be added:

$$\mathcal{G}(D_i, \mathcal{S}_{k-1}) = f(\mathcal{S}_{k-1} \cup D_i) - f(\mathcal{S}_{k-1})$$

Simple investigation through a Venn diagram proof shows that $\mathcal{G}$ can be re-written as

$$\mathcal{G}(D_i, \mathcal{S}_{k-1}) = |D_i \cap (\cup_{j \neq i} D_j) - \mathcal{S}_{k-1}|$$

Let's denote $D_i \cap (\cup_{j \neq i} D_j)$ by $\cap_i$. The following equations prove the theorem.

$$\begin{aligned}
&\mathcal{S}_{k-1} \subseteq \mathcal{S}_k \\
&\mathcal{S}'_{k-1} \supseteq \mathcal{S}'_k \\
&\cap_i \cap \mathcal{S}'_{k-1} \supseteq \cap_i \cap \mathcal{S}'_k \\
&\cap_i - \mathcal{S}_{k-1} \supseteq \cap_i - \mathcal{S}_k \\
&|\cap_i - \mathcal{S}_{k-1}| \geq |\cap_i - \mathcal{S}_k| \\
&\mathcal{G}(D_i, \mathcal{S}_{k-1}) \geq \mathcal{G}(D_i, \mathcal{S}_k) \\
&f(\mathcal{S}_{k-1} \cup D_i) - f(\mathcal{S}_{k-1}) \geq f(\mathcal{S}_k \cup D_i) - f(\mathcal{S}_k)
\end{aligned}$$

Here, $\mathcal{S}'_k$ is the set of all N-grams in the vocabulary that are not present in $\mathcal{S}_k$. The gain of adding a sentence, D_i, to an empty summary is a non-negative value.

$$\mathcal{G}(D_i, \mathcal{S}_0) = C \geq 0$$

By induction, we will get

$$\mathcal{G}(D_i, \mathcal{S}_0) \geq \mathcal{G}(D_i, \mathcal{S}_1) \geq \cdots \geq \mathcal{G}(D_i, \mathcal{S}_k) \geq 0$$

□

Theorem 1 implies the general case of submodularity:

$$\forall m, n, 0 \leq m \leq n \leq |D| \Rightarrow \mathcal{G}(D_i, \mathcal{S}_m) \geq \mathcal{G}(D_i, \mathcal{S}_n)$$

Maximizing this submodular function is an NP-hard problem (Khuller et al., 1999). A common way to solve this maximization problem is to start with an empty set, and in each iteration pick a set that maximizes the gain. It has been shown before in (Kulik et al., 2009) that if f is a submodular, nondecreasing set function and $f(\emptyset) = 0$, then such a greedy algorithm finds a set $\mathcal{S}$, whose gain is at least as high as $(1 - 1/e)$ of the best possible solution. Therefore, we can optimize the keyphrase coverage as described in Algorithm 1.

4 Experimental Setup

We use the annotated data described in Section 2. In summary, the annotation consisted of two parts: nugget extraction and nugget distribution analysis. Five annotators were employed to annotate the sentences in each of the 25 citation summaries and write down the nuggets (non-overlapping contributions) of the target paper. Then using these

Summary generated using bigram-based keyphrases	
ID	Sentence
P06-1048:1	Ziff-Davis Corpus Most previous work (Jing 2000; Knight and Marcu 2002; Riezler et al 2003; Nguyen et al 2004a; Turner and Charniak 2005; McDonald 2006) has relied on automatically constructed parallel corpora for training and evaluation purposes.
J05-4004:18	Between these two extremes, there has been a relatively modest amount of work in sentence simplification (Chandrasekar, Doran, and Bangalore 1996; Mahesh 1997; Carroll et al 1998; Grefenstette 1998; Jing 2000; Knight and Marcu 2002) and document compression (Daume III and Marcu 2002; Daume III and Marcu 2004; Zajic, Dorr, and Schwartz 2004) in which words, phrases, and sentences are selected in an extraction process.
A00-2024:9	The evaluation of sentence reduction (see (Jing, 2000) for details) used a corpus of 500 sentences and their reduced forms in human-written abstracts.
N03-1026:17	To overcome this problem, linguistic parsing and generation systems are used in the sentence condensation approaches of Knight and Marcu (2000) and Jing (2000).
P06-2019:5	Jing (2000) was perhaps the first to tackle the sentence compression problem.

Table 5: Bigram-based summary generated for A00-1043.

Algorithm 1 The greedy algorithm for summary generation

$k \leftarrow$ the number of sentences in the summary
$D_i \leftarrow$ keyphrases in d_i
$\mathcal{S} \leftarrow \emptyset$
for l = 1 to k **do**
  $s_l \leftarrow \arg\max_{D_i \in D} |D_i \cap (\cup_{j \neq i} D_j)|$
  $\mathcal{S} \leftarrow \mathcal{S} \cup s_l$
  for j = 1 to $|D|$ **do**
    $D_j \leftarrow D_j - s_l$
  end for
end for
return $\mathcal{S}$

nugget sets, each sentence was annotated with the nuggets it contains. This results in a sentence-fact matrix that helps with the evaluation of the summary. The summarization goal and the intuition behind the summarizing system is to select a few (5 in our experiments) sentences and cover as many nuggets as possible. Each sentence in a citation summary may contain 0 or more nuggets and not all nuggets are mentioned an equal number of times. Covering some nuggets (contributions) is therefore more important than others and should be weighted highly.

To capture this property, the pyramid score seems the best evaluation metric to use. We use the pyramid evaluation method (Nenkova and Passonneau, 2004) at the sentence level to evaluate the summary created for each set. We benefit from the list of annotated nuggets provided by the annotators as the ground truth of the summarization evaluation. These annotations give the list of nuggets covered by each sentence in each citation summary, which are equivalent to the *summarization content unit (SCU)* as described in (Nenkova and Passonneau, 2004).

The pyramid score for a summary is calculated as follows. Assume a pyramid that has n tiers, T_i, where tier $T_i > T_j$ if $i > j$ (i.e., T_i is not below T_j, and that if a nugget appears in more sentences, it falls in a higher tier.). Tier T_i contains nuggets that appeared in i sentences, and thus has weight i. Suppose $|T_i|$ shows the number of nuggets in tier T_i, and Q_i is the size of a subset of T_i whose members appear in the summary. Further suppose Q shows the sum of the weights of the facts that are covered by the summary. $Q = \sum_{i=1}^{n} i \times Q_i$. In addition, the optimal pyramid score for a summary with X facts, is

$$Max = \sum_{i=j+1}^{n} i \times |T_i| + j \times (X - \sum_{i=j+1}^{n} |T_i|)$$

where $j = \max_i(\sum_{t=i}^{n} |T_t| \geq X)$. The pyramid score for a summary is then calculated as follows.

$$P = \frac{Q}{Max}$$

This score ranges from 0 to 1, and a high score shows the summary contains more heavily weighted facts.

4.1 Baselines and Gold Standards

To evaluate the quality of the summaries generated by the greedy algorithm, we compare its pyramid score in each of the 25 citation summaries with those of a gold standard, a random summary, and four other methods. The gold standards are summaries created manually using 5 sentences. The 5 sentences are manually selected in a way to cover as many nuggets as possible with higher priority for the nuggets with higher frequencies. We also created random summaries using Mead (Radev et al., 2004). These summaries

are basically a random selection of 5 sentences from the pool of sentences in the citation summary. Generally we expect the summaries created by the greedy method to be significantly better than random ones.

In addition to the gold and random summaries, we also used 4 baseline state of the art summarizers: LexRank, the clustering C-RR and C-LexRank, and Maximal Marginal Relevance (MMR). LexRank (Erkan and Radev, 2004) works based on a random walk on the cosine similarity of sentences and prints out the most frequently visited sentences. Said differently, LexRank first builds a network in which nodes are sentences and edges are cosine similarity values. It then uses the eigenvalue centralities to find the most central sentences. For each set, the top 5 sentences on the list are chosen for the summary.

The clustering methods, C-RR and C-LexRank, work by clustering the cosine similarity network of sentences. In such a network, nodes are sentences and edges are cosine similarity of node pairs. Clustering would intuitively put nodes with similar nuggets in the same clusters as they are more similar to each other. The C-RR method as described in (Qazvinian and Radev, 2008) uses a round-robin fashion to pick sentences from each cluster, assuming that the clustering will put the sentences with similar facts into the same clusters. Unlike C-RR, C-LexRank uses LexRank to find the most salient sentences in each cluster, and prints out the most central nodes of each cluster as summary sentences.

Finally, MMR uses the full cosine similarity matrix and greedily chooses sentences that are the least similar to those already selected for the summary (Carbonell and Goldstein, 1998). In particular,

$$MMR = \arg \min_{d_i \in D-A} \Big[\max_{d_j \in A} Sim(d_i, d_j) \Big]$$

where A is the set of sentences in the summary, initially set to $A = \emptyset$. This method is different from ours in that it chooses the least similar sentence to the summary in each iteration.

4.2 Results and Discussion

As mentioned before, we use the text of the abstracts of all the papers in AAN as the background, and each citation set as a separate foreground corpus. For each citation set, we use the method described in Section 3.1 to extract significant N-grams of each sentence. We then use the keyphrase set representation of each sentence to build the summaries using Algorithm 1. For each of the 25 citation summaries, we build 4 different summaries using unigrams, bigrams, 3-grams, and 4-grams respectively. Table 5 shows a 5-sentence summary created using algorithm 1 for the paper A00-1043 (Jing, 2000).

The pyramid scores for different methods are reported in Figure 1 together with the scores of gold standards, manually created to cover as many nuggets as possible in 5 sentences, as well as summary evaluations of the 4 baseline methods described above. This Figure shows how the keyphrase based summarization method when employing N-grams of size 3 or smaller, outperforms other baseline systems significantly. More importantly, Figure 1 also indicates that this method shows more stable results and low variation in summary quality when keyphrases of size 3 or smaller are employed. In contrast, MMR shows high variation in summary qualities making summaries that obtain pyramid scores as low as 0.15.

Another important advantage of this method is that we do not need to calculate the cosine similarity of the pairs of sentences, which would add a running time of $O(|D|^2|V|)$ in the number of documents, $|D|$, and the size of the vocabulary $|V|$ to the algorithm.

5 Conclusion and Future Work

This paper presents a summarization methodology that employs keyphrase extraction to find important contributions of scientific articles. The summarization is based on citation sentences and picks sentences to cover nuggets (represented by keyphrases) or contributions of the target papers. In this setting the best summary would have as few sentences and at the same time as many nuggets as possible. In this work, we use pointwise KL-divergence to extract statistically significant N-grams and use them to represent nuggets. We then apply a new set function for the task of summarizing scientific articles. We have proved that this function is submodular and concluded that a

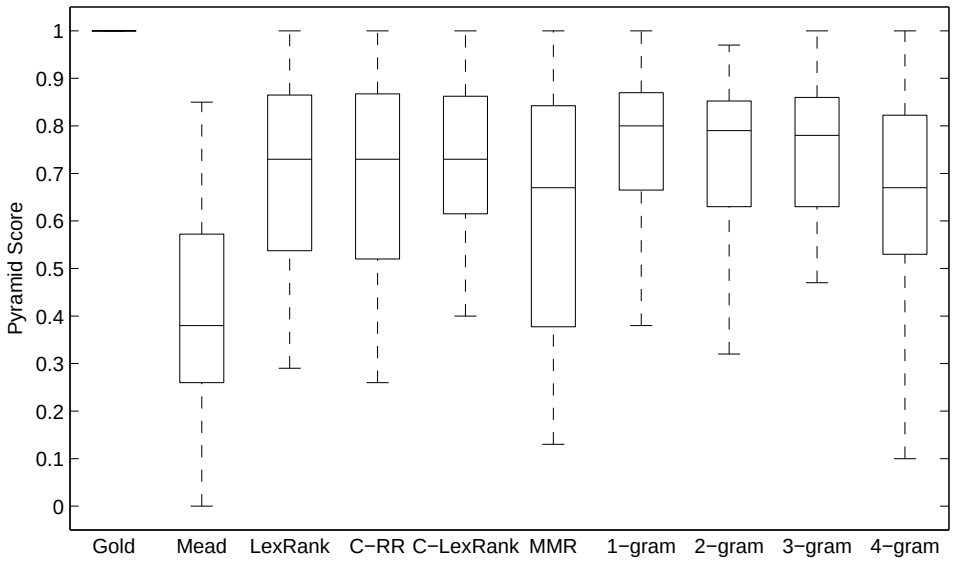


Figure 1: Evaluation Results (summaries with 5 sentences): The median pyramid score over 25 datasets using different methods.

greedy algorithm will result in a near-optimum set of covered nuggets using only 5 sentences. Our experiments in this paper confirm that the summaries created based on the presented algorithm are better than randomly generated summary, and also outperform other state of the art summarization methods in most cases. Moreover, we show how this method generates more stable summaries with lower variation in summary quality when N-grams of size 3 or smaller are employed.

A future direction for this work is to perform post-processing on the summaries and re-generate sentences that cover the extracted nuggets. However, the ultimate goal is to eventually develop systems that can produce summaries of entire research areas, summaries that will enable researchers to easily and quickly switch between fields of research.

One future study that will help us generate better summaries is to understand how nuggets are generated by authors. In fact, modeling the nugget coverage behavior of paper authors will help us identify more important nuggets and discover some aspects of the paper that would otherwise be too difficult by just reading the paper itself.

6 Acknowledgements

This work is in part supported by the National Science Foundation grant “iOPENER: A Flexible Framework to Support Rapid Learning in Unfamiliar Research Domains”, jointly awarded to University of Michigan and University of Maryland as IIS 0705832, and in part by the NIH Grant U54 DA021519 to the National Center for Integrative Biomedical Informatics.

Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the supporters.

References

Bradshaw, Shannon. 2002. *Reference Directed Indexing: Indexing Scientific Literature in the Context of Its Use*. Ph.D. thesis, Northwestern University.

Bradshaw, Shannon. 2003. Reference directed indexing: Redeeming relevance for subject search in citation indexes. In *Proceedings of the 7th European*

Conference on Research and Advanced Technology for Digital Libraries.

Carbonell, Jaime G. and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *SIGIR'98*, pages 335–336.

Clarkson, PR and R Rosenfeld. 1997. Statistical language modeling using the cmu-cambridge toolkit. *Proceedings ESCA Eurospeech*, 47:45–148.

Elkiss, Aaron, Siwei Shen, Anthony Fader, Güneş Erkan, David States, and Dragomir R. Radev. 2008. Blind men and elephants: What do citation summaries tell us about a research article? *Journal of the American Society for Information Science and Technology*, 59(1):51–62.

Erkan, Güneş and Dragomir R. Radev. 2004. Lexrank: Graph-based centrality as salience in text summarization. *Journal of Artificial Intelligence Research*.

Jing, Hongyan. 2000. Sentence reduction for automatic text summarization. In *Proceedings of the sixth conference on Applied natural language processing*, pages 310–315, Morristown, NJ, USA. Association for Computational Linguistics.

Kan, Min-Yen, Judith L. Klavans, and Kathleen R. McKeown. 2002. Using the Annotated Bibliography as a Resource for Indicative Summarization. In *Proceedings of LREC 2002*, Las Palmas, Spain.

Khuller, Samir, Anna Moss, and Joseph (Seffi) Naor. 1999. The budgeted maximum coverage problem. *Inf. Process. Lett.*, 70(1):39–45.

Kulik, Ariel, Hadas Shachnai, and Tami Tamir. 2009. Maximizing submodular set functions subject to multiple linear constraints. In *SODA '09*, pages 545–554.

Kupiec, Julian, Jan Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *SIGIR '95*, pages 68–73, New York, NY, USA. ACM.

Manning, Christopher D. and Hirich Schutze. 2002. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, London, England.

Mei, Qiaozhu and ChengXiang Zhai. 2008. Generating impact-based summaries for scientific literature. In *Proceedings of ACL '08*, pages 816–824.

Mohammad, Saif, Bonnie Dorr, Melissa Egan, Ahmed Hassan, Pradeep Muthukrishan, Vahed Qazvinian, Dragomir Radev, and David Zajic. 2009. Using citations to generate surveys of scientific paradigms. In *NAACL 2009*, pages 584–592, June.

Nanba, Hidetsugu and Manabu Okumura. 1999. Towards multi-paper summarization using reference information. In *IJCAI1999*, pages 926–931.

Nanba, Hidetsugu, Noriko Kando, and Manabu Okumura. 2004. Classification of research papers using citation links and citation types: Towards automatic review article generation. In *Proceedings of the 11th SIG Classification Research Workshop*, pages 117–134, Chicago, USA.

Nenkova, Ani and Rebecca Passonneau. 2004. Evaluating content selection in summarization: The pyramid method. *Proceedings of the HLT-NAACL conference*.

Qazvinian, Vahed and Dragomir R. Radev. 2008. Scientific paper summarization using citation summary networks. In *COLING 2008*, Manchester, UK.

Radev, Dragomir, Timothy Allison, Sasha Blair-Goldensohn, John Blitzer, Arda Çelebi, Stanko Dimitrov, Elliott Drabek, Ali Hakim, Wai Lam, Danyu Liu, Jahna Otterbacher, Hong Qi, Horacio Saggion, Simone Teufel, Michael Topper, Adam Winkel, and Zhu Zhang. 2004. MEAD - a platform for multidocument multilingual text summarization. In *LREC 2004*, Lisbon, Portugal, May.

Radev, Dragomir R., Pradeep Muthukrishnan, and Vahed Qazvinian. 2009. The ACL anthology network corpus. In *ACL workshop on Natural Language Processing and Information Retrieval for Digital Libraries*.

Siddharthan, Advaith and Simone Teufel. 2007. Whose idea was this, and why does it matter? attributing scientific work to citations. In *Proceedings of NAACL/HLT-07*.

Teufel, Simone. 2005. Argumentative Zoning for Improved Citation Indexing. *Computing Attitude and Affect in Text: Theory and Applications*, pages 159–170.

Tomokiyo, Takashi and Matthew Hurst. 2003. A language model approach to keyphrase extraction. In *Proceedings of the ACL 2003 workshop on Multiword expressions*, pages 33–40.

2D Trie for Fast Parsing

Xian Qian, Qi Zhang, Xuanjing Huang, Lide Wu
Institute of Media Computing
School of Computer Science, Fudan University
{xianqian, qz, xjhuang, ldwu}@fudan.edu.cn

Abstract

In practical applications, decoding speed is very important. Modern structured learning technique adopts template based method to extract millions of features. Complicated templates bring about abundant features which lead to higher accuracy but more feature extraction time. We propose Two Dimensional Trie (2D Trie), a novel efficient feature indexing structure which takes advantage of relationship between templates: feature strings generated by a template are prefixes of the features from its extended templates. We apply our technique to Maximum Spanning Tree dependency parsing. Experimental results on Chinese Tree Bank corpus show that our 2D Trie is about 5 times faster than traditional Trie structure, making parsing speed 4.3 times faster.

1 Introduction

In practical applications, decoding speed is very important. Modern structured learning technique adopts template based method to generate millions of features. Such as shallow parsing (Sha and Pereira, 2003), named entity recognition (Kazama and Torisawa,), dependency parsing (McDonald et al., 2005), etc.

The problem arises when the number of templates increases, more features generated, making the extraction step time consuming. Especially for maximum spanning tree (MST) dependency parsing, since feature extraction requires quadratic time even using a first order model. According to Bohnet's report (Bohnet, 2009), a fast feature extraction beside of a fast parsing algorithm is important for the parsing and training speed. He takes 3 measures for a 40X speedup, despite the same inference algorithm. One important measure is to store the feature vectors in file to skip feature extraction, otherwise it will be the bottleneck.

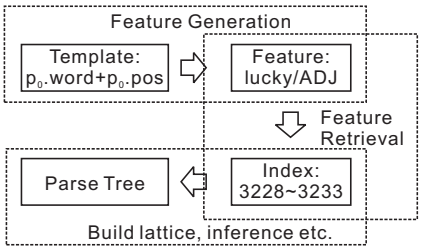


Figure 1: Flow chart of dependency parsing. p_0.*word*, p_0.*pos* denotes the word and POS tag of parent node respectively. Indexes correspond to the features conjoined with dependency types, e.g., *lucky/ADJ/OBJ*, *lucky/ADJ/NMOD*, etc.

Now we quickly review the feature extraction stage of structured learning. Typically, it consists of 2 steps. First, features represented by strings are generated using templates. Then a feature indexing structure searches feature indexes to get corresponding feature weights. Figure 1 shows the flow chart of MST parsing, where p_0.*word*, p_0.*pos* denote the word and POS tag of parent node respectively.

We conduct a simple experiment to investigate decoding time of MSTParser, a state-of-the-art java implementation of dependency parsing [1]. Chinese Tree Bank 6 (CTB6) corpus (Palmer and

[1] http://sourceforge.net/projects/mstparser

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 904–912,
Beijing, August 2010

Step	Feature Generation	Index Retrieval	Other	Total
Time	300.27	61.66	59.48	421.41

Table 1: Time spent of each step (seconds) of MSTParser on CTB6 standard test data (2660 sentences). Details of the hardware and corpus are described in section 5

Xue, 2009) with standard train/development/test split is used for evaluation. Experimental results are shown in Table 1. The observation is that time spent of inference is trivial compared with feature extraction. Thus, speeding up feature extraction is critical especially when large template set is used for high accuracy.

General indexing structure such as Hash and Trie does not consider the relationships between templates, therefore they could not speed up feature generation, and are not completely efficient for searching feature indexes. For example, feature string s_1 generated by template "p_0.*word*" is prefix of feature s_2 from template "p_0.*word* + c_0.*word*" (word pair of parent and child), hence index of s_1 could be used for searching s_2. Further more, if s_1 is not in the feature set, then s_2 must be absent, its generation can be skipped.

We propose Two Dimensional Trie (2D Trie), a novel efficient feature indexing structure which takes advantage of relationship between templates. We apply our technique to Maximum Spanning Tree dependency parsing. Experimental results on CTB6 corpus show that our 2D Trie is about 5 times faster than traditional Trie structure, making parsing speed 4.3 times faster.

The paper is structured as follows: in section 2, we describe template tree which represents relationship between templates; in section 3, we describe our new 2D Trie structure; in section 4, we analyze the complexity of the proposed method and general string indexing structures for parsing; experimental results are shown in section 5; we conclude the work in section 6.

2 Template tree

2.1 Formulation of template

A template is a set of template units which are manually designed: $T = \{t_1, \ldots, t_m\}$. For convenience, we use another formulation: $T = t_1 + \ldots + t_m$. All template units appearing in this paper are described in Table 2, most of them are widely used. For example, "T = p_0.*word* + c_0.*word*$|^l$ " denotes the word pair of parent and child nodes, conjoined with their distance.

Unit	Meaning
p_{-i}/p_i	the i^{th} node left/right to parent node
c_{-i}/c_i	the i^{th} node left/right to child node
r_{-i}/r_i	the i^{th} node left/right to root node
n.*word*	word of node n
n.*pos*	POS tag of node n
n.*length*	word length of node n
$\|^l$	conjoin current feature with linear distance between child node and parent node
$\|_d$	conjoin current feature with direction of dependency (left/right)

Table 2: Template units appearing in this paper

2.2 Template tree

In the rest of the paper, for simplicity, let s_i be a feature string generated by template T_i.

We define the relationship between templates: T_1 is the **ancestor** of T_2 if and only $T_1 \subset T_2$, and T_2 is called the **descendant** of T_1. Recall that, feature string s_1 is prefix of feature s_2. Suppose $T_3 \subset T_1 \subset T_2$, obviously, the most efficient way to look up indexes of s_1, s_2, s_3 is to search s_3 first, then use its index id_3 to search s_1, and finally use id_1 to search s_2. Hence the relationship between T_2 and T_3 can be neglected.

Therefore we define **direct ancestor** of T_1: T_2 is a direct ancestor of T_1 if $T_2 \subset T_1$, and there is no template T' such that $T_2 \subset T' \subset T_1$. Correspondingly, T_1 is called the **direct descendant** of T_2.

Template graph $G = (V, E)$ is a directed graph that represents the relationship between templates, where $V = \{T_1, \ldots, T_n\}$ is the template set, $E = \{e_1, \ldots, e_N\}$ is the edge set. Edge from T_i to T_j exists, if and only if T_i is the direct ancestor of T_j. For templates having no ancestor, we add an empty template as their common direct ancestor, which is also the root of the graph.

The left part of Figure 2 shows a template graph for templates T_1 =p_0.*word*, T_2 =p_0.*pos* , T_3 =p_0.*word* + p_0.*pos*. In this example, T_3 has 2 direct ancestors, but in fact s_3 has only one prefix

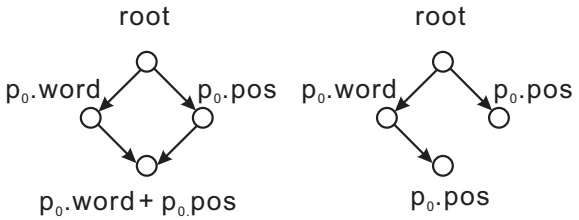


Figure 2: Left graph shows template graph for T_1 =p0.*word*, T_2 =p0.*pos* , T_3 =p0.*word* + p0.*pos*. Right graph shows the corresponding template tree, where each vertex saves the subset of template units that do not belong to its father

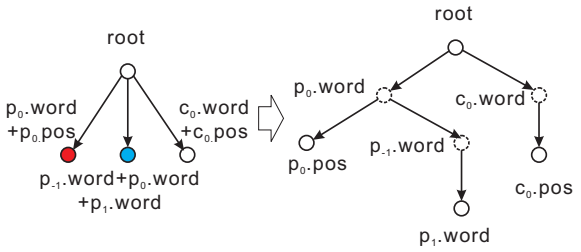


Figure 3: Templates that are partially overlapped: $T_{red} \cap T_{blue}$ =p0.*word*, virtual vertexes shown in dashed circle are created to extract the common unit

which depends on the order of template units in generation step. If $s_3 = s_1 + s_2$, then its prefix is s_1, otherwise its prefix is s_2. In this paper, we simply use the breadth-first tree of the graph for disambiguation, which is called **template tree**. The only direct ancestor T_1 of T_2 in the tree is called **father** of T_2, and T_2 is a **child** of T_1. The right part of Figure 2 shows the corresponding template tree, where each vertex saves the subset of template units that do not belong to its father.

2.3 Virtual vertex

Consider the template tree in the left part of Figure 3, red vertex and blue vertex are partially overlapped, their intersection is p0.*word*, if string s from template T =p0.*word* is absent in feature set, then both nodes can be neglected. For efficiently pruning candidate templates, each vertex in template tree is restricted to have exactly one template unit (except root). Another important reason for such restriction will be given in the next section.

To this end, virtual vertexes are created for multi-unit vertexes. For efficient pruning, the new virtual vertex should extract the most common template unit. A natural goal is to minimize the creation number. Here we use a simple greedy strategy, for the vertexes sharing a common father, the most frequent common unit is extracted as new vertex. Virtual vertexes are iteratively created in this way until all vertexes have one unit. The final template tree is shown in the right part of Figure 3, newly created virtual vertexes are shown in dashed circle.

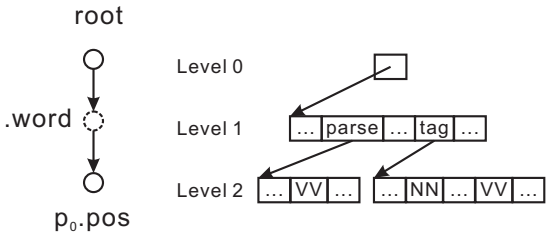


Figure 4: 2D Trie for single template, alphabets at level 1 and level 2 are the word set, POS tag set respectively

3 2D Trie

3.1 Single template case

Trie stores strings over a fixed alphabet, in our case, feature strings are stored over several alphabets, such as word list, POS tag list, etc. which are extracted from training corpus.

To illustrate 2D Trie clearly, we first consider a simple case, where only one template used. The template tree degenerates to a sequence, we could use a Trie like structure for feature indexing, the only difference from traditional Trie is that nodes at different levels could have different alphabets. One example is shown in Figure 4. There are 3 feature strings from template "p0.*word* + p0.*pos*": {*parse/VV, tag/VV, tag/VV*}. Alphabets at level 1 and level 2 are the word set, POS tag set respectively, which are determined by corresponding template vertexes.

As mentioned before, each vertex in template tree has exactly one template unit, therefore, at each level, we look up an index of a word or POS

He	PRP	2648	21
had	VBD	2731	27
been	VBN	1121	28
a	DT	0411	04
sales	NNS	5064	13
and	CC	0631	01
marketing	NN	3374	12
executive	NN	1923	12
with	IN	6023	06
Chrysler	NNP	1560	13
for	IN	2203	06
20	CD	0056	02
years	NNS	6778	14

Figure 5: Look up indexes of words and POS tags beforehand.

tag in sentence, not their combinations. Hence the number of alphabets is limited, and all the indexes could be searched beforehand for reuse, as shown in Figure 5, the token table is converted to a index table. For example, when generating features at position i of a sentence, template "r_0.*word* + r_1.*word*" requires index of $i+1^{th}$ word in the sentence, which could be reused for generation at position $i+1$.

3.2 General case

Generally, for vertex in template tree with K children, children of corresponding Trie node are arranged in a matrix of K rows and L columns, L is the size of corresponding alphabet. If the vertex is not virtual, i.e., it generates features, one more row is added at the bottom to store feature indexes. Figure 6 shows the 2D Trie for a general template tree.

3.3 Feature extraction

When extracting features for a pair of nodes in a sentence, template tree and 2D Trie are visited in breath first traversal order. Each time, an alphabet and a token index j from index table are selected according to current vertex. For example, POS tag set and the index of the POS tag of parent node are selected as alphabet and token index respectively for vertex "p_0.*pos*". Then children in the j^{th} column of the Trie node are visited, valid children and corresponding template vertexes are saved for further retrieval or generate feature indexes if the child is at the bottom and current Trie node is not virtual. Two queues are maintained to

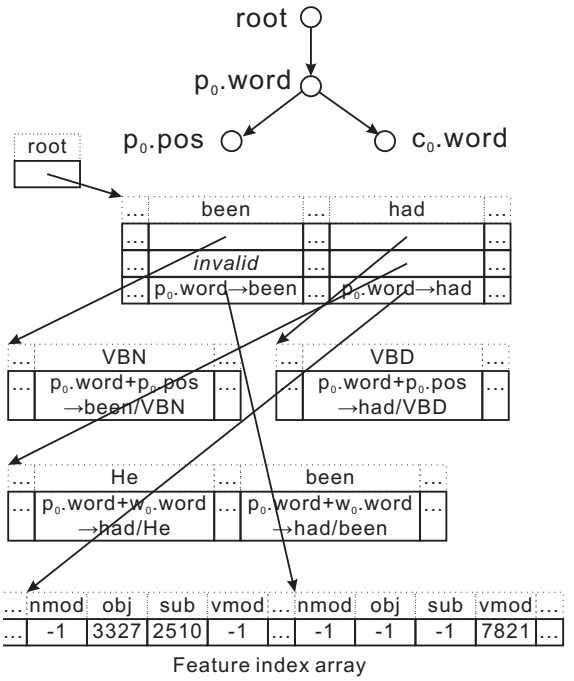


Figure 6: 2D trie for a general template tree. Dashed boxes are keys of columns, which are not stored in the structure

save the valid children and Trie nodes. Details of feature extraction algorithm are described in Algorithm 1.

3.4 Implementation

When feature set is very large, space complexity of 2D Trie is expensive. Therefore, we use Double Array Trie structure (Aoe, 1989) for implementation. Since children of 2D Trie node are arranged in a matrix, not an array, so each element of the base array has a list of bases, not one base in standard structure. For children that store features, corresponding bases are feature indexes. One example is shown in Figure 7. The root node has 3 bases that point to three rows of the child matrix of vertex "p_0.*word*" respectively. Number of bases in each element need not to be stored, since it can be obtained from template vertex in extraction procedure.

Building algorithm is similarly to Double Array Trie, when inserting a Trie node, each row of the child matrix is independently insert into base and check arrays using brute force strategy. The inser-

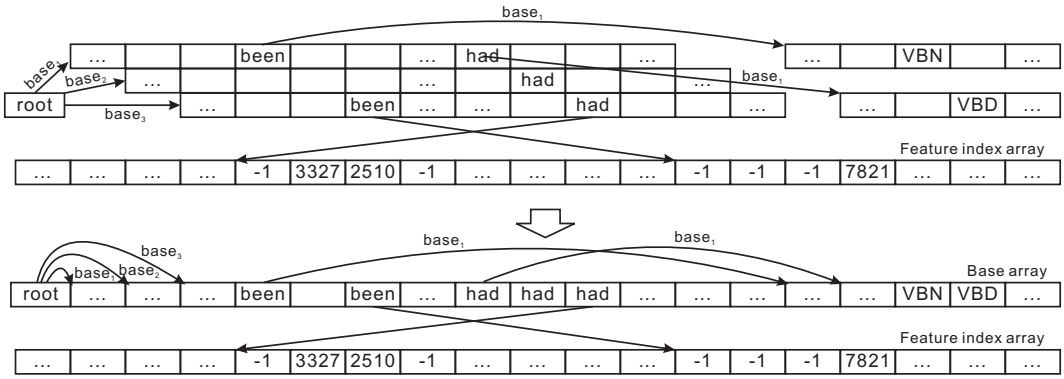


Figure 7: Build base array for 2D Trie in Figure 6. String in the box represents the key of the child. Blank boxes are the invalid children. The root node has 3 bases that point to three rows of the child matrix of vertex "p_0.*word*" respectively

Algorithm 1 Feature extraction using 2D Trie

Input: 2D Trie that stores features, template tree, template graph, a table storing token indexes, parent and child positions

Output: Feature index set S of dependency from parent to child.

Create template vertex queue Q_1 and Trie node queue Q_2. Push roots of template tree and Trie into Q_1, Q_2 respectively. $S = \emptyset$
while Q_1 is not empty, **do**
  Pop a template vertex T from Q_1 and a Trie node N from Q_2. Get token index j from index table according to T.
  for $i = 1$ to child number of T
    if child of N at row i column j is valid, push it into Q_2 and push the i^{th} child of T into Q_1.
    else
      remove decedents of i^{th} child of T from template tree
    end if
  end for
  if T is not virtual and the last child of N in column j is valid
      Enumerate dependency types, add valid feature indexes to S
  end if
end while
Return S.

tion repeats recursively until all features stored.

4 Complexity analysis

Let

- $|T|$ = number of templates
- $|t|$ = number of template units
- $|V|$ = number of vertexes in template tree, i.e, $|t|$+ number of virtual vertexes
- $|F|$ = number of features
- l = length of sentence
- $|f|$ = average length of feature strings

The procedure of 2D Trie for feature extraction consists of 2 steps: tokens in string table are mapped to their indexes, then Algorithm 1 is carried out for all node pairs of sentence. In the first step, we use double array Trie for efficient mapping. In fact, time spent is trivial compared with step 2 even by binary search. The main time spent of Algorithm 1 is the traversal of the whole template tree, in the worst case, no vertexes removed, so the time complexity of a sentence is $l^2|V|$, which is proportional to $|V|$. In other words, minimizing the number of virtual vertexes is important for efficiency.

For other indexing structures, feature generation is a primary step of retrieval. For each node

Structure	Generation	Retrieval
2D Trie	$l^2\|V\|$	
Hash / Trie	$l^2\|t\|$	$l^2\|f\|\|T\|$
Binary Search	$l^2\|t\|$	$l^2\|T\|\log\|F\|$

Table 3: Time complexity of different indexing structures.

pair of sentence, $|t|$ template units are processed, including concatenations of tokens and split symbols (split tokens in feature strings), boundary check (e.g, p_{-1}.*word* is out of boundary for beginning node of sentence). Thus the generation requires $l^2|t|$ processes. Notice that, time spent of each process varies on the length of tokens.

For feature string s with length $|s|$, if perfect hashing technique is adopted for index retrieval, it takes $|s|$ calculations to get hash value and a string comparison to check the string at the calculated position. So the time complexity is proportional to $|s|$, which is the same as Trie. Hence the total time for a sentence is $l^2|f||T|$. If binary search is used instead, $\log|F|$ string comparisons are required, complexity for a sentence is $l^2|T|\log|F|$.

Time complexity of these structures is summarized in Table 3.

5 Experiments

5.1 Experimental settings

We use Chinese Tree Bank 6.0 corpus for evaluation. The constituency structures are converted to dependency trees by Penn2Malt [2] toolkit and the standard training/development/test split is used. 257 sentences that failed in the conversion were removed, yielding 23316 sentences for training, 2060 sentences for development and 2660 sentences for testing respectively.

Since all the dependency trees are projective, a first order projective MST parser is naturally adopted. Online Passive Aggressive algorithm (Crammer et al., 2006) is used for fast training, 2 parameters, i.e, iteration number and C, are tuned on development data. The quality of the parser is measured by the labeled attachment score (LAS), i.e., the percentage of tokens with correct head and dependency type.

[2]http://w3.msi.vxu.se/ nivre/research/Penn2Malt.html

Group	IDs	#Temp.	#Vert.	#Feat.	LAS
1	1-2	72	91	$3.23M$	79.55%
2	1-3	128	155	$10.4M$	81.38%
3	1-4	240	275	$25.0M$	81.97%
4	1-5	332	367	$34.8M$	82.44%

Table 5: Parsing accuracy and number of templates, vertexes in template tree, features in decoding stage (zero weighted features are excluded) of each group.

We compare the proposed structure with Trie and binary search. We do not compare with perfect hashing, because it has the same complexity as Trie, and is often used for large data base retrieval, since it requires only one IO operation. For easy comparison, all feature indexing structures and the parser are implemented with C++. All experiments are carried out on a 64bit linux platform (CPU: Intel(R) Xeon(R) E5405, 2.00GHz, Memory: 16G Bytes). For each template set, we run the parser five times on test data and the averaged parsing time is reported.

5.2 Parsing speed comparison

To investigate the scalability of our method, rich templates are designed to generate large feature sets, as shown in Table 4. All templates are organized into 4 groups. Each row of Table 5 shows the details of a group, including parsing accuracy and number of templates, vertexes in template tree, and features in decoding stage (zero weighted features are excluded).

There is a rough trend that parsing accuracy increases as more templates used. Though such trend is not completely correct, the clear conclusion is that, abundant templates are necessary for accurate parsing.

Though algorithm described in section 2.3 for minimizing the number of virtual vertexes is heuristic, empirical results are satisfactory, number of newly created vertexes is only 10% as original templates. The reason is that complex templates are often extended from simple ones, their differences are often one or two template units.

Results of parsing time comparison are shown in Table 6. We can see that though time complexity of dynamic programming is cubic, parsing time of all systems is consistently dominated

ID	Templates		
1	$p_i.word$	$p_i.pos$	$p_i.word$+$p_i.pos$
	$c_i.word$	$c_i.pos$	$c_i.word$+$c_i.pos$ $(\|i\| \leq 2)$
	$p_i.length$	$p_i.length$+$p_i.pos$	
	$c_i.length$	$c_i.length$+$c_i.pos$	$(\|i\| \leq 1)$
	$p_0.length$+$c_0.length\|_d^l$	$p_0.length$+$c_0.length$+$c_0.pos\|_d^l$	$p_0.length$+$p_0.pos$+$c_0.length\|_d^l$
	$p_0.length$+$p_0.pos$+$c_0.pos\|_d^l$	$p_0.pos$+$c_0.length$+$c_0.pos\|_d^l$	$p_0.length$+$p_0.pos$+$c_0.length$+$c_0.pos\|_d^l$
	$p_i.length$+$p_j.length$+$c_k.length$+$c_m.length\|_d^l$		$(\|i\| + \|j\| + \|k\| + \|m\| \leq 2)$
	$r_0.word$	$r_{-1}.word$+$r_0.word$	$r_0.word$+$r_1.word$
	$r_0.pos$	$r_{-1}.pos$+$r_0.pos$	$r_0.pos$+$r_1.pos$
2	$p_i.pos$+$c_j.pos\|_d$	$p_i.word$+$c_j.word\|_d$	$p_i.pos$+$c_j.word$+$c_j.pos\|_d$
	$p_i.word$+$p_i.pos$+$c_j.pos\|_d$	$p_i.word$+$p_i.pos$+$c_j.word\|_d$	$p_i.word$+$c_j.word$+$c_j.pos\|_d$
	$p_i.word$+$p_i.pos$+$c_j.word$+$c_j.pos\|_d$		$(\|i\| + \|j\| = 0)$
	Conjoin templates in the row above with $\|^l$		
3	Similar with 2 $\|i\| + \|j\| = 1$		
4	Similar with 2 $\|i\| + \|j\| = 2$		
5	$p_i.word$ + $p_j.word$ + $c_k.word\|_d$	$p_i.word$ + $c_j.word$ + $c_k.word\|_d$	
	$p_i.pos$ + $p_j.pos$ + $c_k.pos\|_d$	$p_i.pos$ + $c_j.pos$ + $c_k.pos\|_d$	$(\|i\| + \|j\| + \|k\| \leq 2)$
	Conjoin templates in the row above with $\|^l$		
	$p_i.word$ + $p_j.word$ + $p_k.word$ + $c_m.word\|_d$		$p_i.word$ + $p_j.word$ + $c_k.word$ + $c_m.word\|_d$
	$p_i.word$ + $c_j.word$ + $c_k.word$ + $c_m.word\|_d$		
	$p_i.pos$ + $p_j.pos$ + $p_k.pos$ + $c_m.pos\|_d$		$p_i.pos$ + $p_j.pos$ + $c_k.pos$ + $c_m.pos\|_d$
	$p_i.pos$ + $c_j.pos$ + $c_k.pos$ + $c_m.pos\|_d$		$(\|i\| + \|j\| + \|k\| + \|m\| \leq 2)$
	Conjoin templates in the row above with $\|^l$		

Table 4: Templates used in Chinese dependency parsing.

by feature extraction. When efficient indexing structure adopted, i.e, Trie or Hash, time index retrieval is greatly reduced, about 4-5 times faster than binary search. However, general structures search features independently, their results could not guide feature generation. Hence, feature generation is still time consuming. The reason is that processing each template unit includes a series of steps, much slower than one integer comparison in Trie search.

On the other hand, 2D Trie greatly reduces the number of feature generations by pruning the template graph. In fact, no string concatenation occurs when using 2D Trie, since all tokens are converted to indexes beforehand. The improvement is significant, 2D Trie is about 5 times faster than Trie on the largest feature set, yielding 13.4 sentences per second parsing speed, about 4.3 times faster.

Space requirement of 2D Trie is about 2.1 times as binary search, and 1.7 times as Trie. One possible reason is that column number of 2D Trie (e.g. size of words) is much larger than standard double array Trie, which has only 256 children, i.e, range of a byte. Therefore, inserting a 2D Trie node is more strict, yielding sparser double arrays.

5.3 Comparison against state-of-the-art

Recent works on dependency parsing speedup mainly focus on inference, such as expected linear time non-projective dependency parsing (Nivre, 2009), integer linear programming (ILP) for higher order non-projective parsing (Martins et al., 2009). They achieve 0.632 seconds per sentence over several languages. On the other hand, Goldberg and Elhadad proposed splitSVM (Goldberg and Elhadad, 2008) for fast low-degree polynomial kernel classifiers, and applied it to transition based parsing (Nivre, 2003). They achieve 53 sentences per second parsing speed on English corpus, which is faster than our results, since transition based parsing is linear time, while for graph based method, complexity of feature extraction is quadratic. Xavier Lluís et al. (Lluís et al., 2009) achieve 8.07 seconds per sentence speed on CoNLL09 (Hajič et al., 2009) Chinese Tree Bank test data with a second order graphic model. Bernd Bohnet (Bohnet, 2009) also uses second order model, and achieves 610 minutes on CoNLL09 English data (2399 sentences, 15.3 second per sentence). Although direct comparison of parsing time is difficult due to the differences in data, models, hardware and implementations,

Group	Structure	Total	Generation	Retrieval	Other	Memory	sent/sec
1	Trie	87.39	63.67	10.33	13.39	$402M$	30.44
	Binary Search	127.84	62.68	51.52	13.64	$340M$	20.81
	2D Trie	39.74	26.29		13.45	$700M$	66.94
2	Trie	264.21	205.19	39.74	19.28	$1.3G$	10.07
	Binary Search	430.23	212.50	198.72	19.01	$1.2G$	6.18
	2D Trie	72.81	53.95		18.86	$2.5G$	36.53
3	Trie	620.29	486.40	105.96	27.93	$3.2G$	4.29
	Binary Search	982.41	484.62	469.44	28.35	$2.9G$	2.71
	2D Trie	146.83	119.56		27.27	$5.9G$	18.12
4	Trie	854.04	677.32	139.70	37.02	$4.9G$	3.11
	Binary Search	1328.49	680.36	609.70	38.43	$4.1G$	2.00
	2D Trie	198.31	160.38		37.93	$8.6G$	13.41

Table 6: Parsing time of 2660 sentences (seconds) on a 64bit linux platform (CPU: Intel(R) Xeon(R) E5405, 2.00GHz, Memory: 16G Bytes). Title "Generation" and "Retrieval" are short for feature generation and feature index retrieval steps respectively.

System	sec/sent
(Martins et al., 2009)	0.63
(Goldberg and Elhadad, 2008)	0.019
(Lluís et al., 2009)	8.07
(Bohnet, 2009)	15.3
(Galley and Manning, 2009)	15.6
ours group1	0.015
ours group2	0.027
ours group3	0.055
ours group4	0.075

Table 7: Comparison against state of the art, direct comparison of parsing time is difficult due to the differences in data, models, hardware and implementations.

these results demonstrate that our structure can actually result in a very fast implementation of a parser. Moreover, our work is orthogonal to others, and could be used for other learning tasks.

6 Conclusion

We proposed 2D Trie, a novel feature indexing structure for fast template based feature extraction. The key insight is that feature strings generated by a template are prefixes of the features from its extended templates, hence indexes of searched features can be reused for further extraction. We applied 2D Trie to dependency parsing task, experimental results on CTB corpus demonstrate the advantages of our technique, about 5 times faster than traditional Trie structure, yielding parsing speed 4.3 times faster, while using only 1.7 times as much memory.

7 Acknowledgments

The author wishes to thank the anonymous reviewers for their helpful comments. This work was partially funded by 973 Program (2010CB327906), The National High Technology Research and Development Program of China (2009AA01A346), Shanghai Leading Academic Discipline Project (B114), Doctoral Fund of Ministry of Education of China (200802460066), and Shanghai Science and Technology Development Funds (08511500302).

References

Aoe, Jun'ichi. 1989. An efficient digital search algorithm by using a double-array structure. *IEEE Transactions on software andengineering*, 15(9):1066–1077.

Bohnet, Bernd. 2009. Efficient parsing of syntactic and semantic dependency structures. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 67–72, Boulder, Colorado, June. Association for Computational Linguistics.

Crammer, Koby, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. In *JMLR 2006*.

Galley, Michel and Christopher D. Manning. 2009. Quadratic-time dependency parsing for machine translation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 773–781, Suntec, Singapore, August. Association for Computational Linguistics.

Goldberg, Yoav and Michael Elhadad. 2008. splitsvm: Fast, space-efficient, non-heuristic, polynomial kernel computation for nlp applications. In *Proceedings of ACL-08: HLT, Short Papers*, pages 237–240, Columbus, Ohio, June. Association for Computational Linguistics.

Hajič, Jan, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 1–18, Boulder, Colorado, June. Association for Computational Linguistics.

Kazama, Jun'ichi and Kentaro Torisawa. A new perceptron algorithm for sequence labeling with non-local features. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 315–324.

Lluís, Xavier, Stefan Bott, and Lluís Màrquez. 2009. A second-order joint eisner model for syntactic and semantic dependency parsing. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 79–84, Boulder, Colorado, June. Association for Computational Linguistics.

Martins, Andre, Noah Smith, and Eric Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 342–350, Suntec, Singapore, August. Association for Computational Linguistics.

McDonald, Ryan, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 91–97. Association for Computational Linguistics.

Nivre, Joakim. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 11th International Conference on Parsing Techniques*, pages 149–160.

Nivre, Joakim. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 351–359, Suntec, Singapore, August. Association for Computational Linguistics.

Palmer, Martha and Nianwen Xue. 2009. Adding semantic roles to the Chinese Treebank. *Natural Language Engineering*, 15(1):143–172.

Sha, Fei and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 134–141, May.

The Bag-of-Opinions Method for Review Rating Prediction from Sparse Text Patterns

Lizhen Qu
Max-Planck Institute for Informatics
lqu@mpii.mpg.de

Georgiana Ifrim
Bioinformatics Research Centre
ifrim@birc.au.dk

Gerhard Weikum
Max-Planck Institute for Informatics
weikum@mpii.mpg.de

Abstract

The problem addressed in this paper is to predict a user's numeric rating in a product review from the text of the review. Unigram and n-gram representations of text are common choices in opinion mining. However, unigrams cannot capture important expressions like "could have been better", which are essential for prediction models of ratings. N-grams of words, on the other hand, capture such phrases, but typically occur too sparsely in the training set and thus fail to yield robust predictors. This paper overcomes the limitations of these two models, by introducing a novel kind of bag-of-opinions representation, where an opinion, within a review, consists of three components: a root word, a set of modifier words from the same sentence, and one or more negation words. Each opinion is assigned a numeric score which is learned, by ridge regression, from a large, domain-independent corpus of reviews. For the actual test case of a domain-dependent review, the review's rating is predicted by aggregating the scores of all opinions in the review and combining it with a domain-dependent unigram model. The paper presents a constrained ridge regression algorithm for learning opinion scores. Experiments show that the bag-of-opinions method outperforms prior state-of-the-art techniques for review rating prediction.

1 Introduction

1.1 Motivation

Opinion mining and sentiment analysis has become a hot research area (Pang and Lee, 2008). There is ample work on analyzing the sentiments of online-review communities where users comment on products (movies, books, consumer electronics, etc.), implicitly expressing their *opinion polarities* (positive, negative, neutral), and also provide numeric *ratings* of products (Titov and McDonald, 2008b; Lerman et al., 2009; Hu and Liu, 2004; Titov and McDonald, 2008a; Pang and Lee, 2005; Popescu and Etzioni, 2005a). Although ratings are more informative than polarities, most prior work focused on classifying text fragments (phrases, sentences, entire reviews) by polarity. However, a product receiving mostly 5-star reviews exhibits better customer purchase behavior compared to a product with mostly 4-star reviews. In this paper we address the learning and prediction of numerical ratings from review texts, and we model this as a metric *regression* problem over an appropriately defined feature space.

Formally, the input is a set of rated documents (i.e., reviews), $\{\mathbf{x}_i, y_i\}_{i=1}^{N}$, where $\mathbf{x}_i$ is a sequence of word-level unigrams $(w_1, ..., w_l)$ and $y_i \in \mathbb{R}$ is a rating. The goal is to learn a function $f(\mathbf{x})$ that maps the word vector $\mathbf{x}$ into a numerical rating $\hat{y}$, which indicates both the polarity and strength of the opinions expressed in a document.

Numerical review rating prediction is harder than classifying by polarity. Consider the following example from Amazon book reviews:

The organization of the book is hard to follow and the chapter titles are not very helpful, so going back and trying to find information is quite

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 913–921,
Beijing, August 2010

difficult.

We note that there are many subjective words (*hard, helpful, difficult*) modified by opinion modifiers such as (*very, quite*) and negation words like (*not*). For rating prediction, considering opinion modifiers is crucial; *very helpful* is a much stronger sentiment than *helpful*. Negation words also need attention. As pointed out by Liu and Seneff (2009) we cannot simply reverse the polarity. For example, if we assign a higher positive score to *very helpful* than to *helpful*, simply reversing the sign of the scores would incorrectly suggest that *not helpful* is less negative than *not very helpful*.

The widely used unigram (bag-of-words) model (Pang and Lee, 2005; Snyder and Barzilay, 2007; Goldberg and Zhu, 2006; Ganu et al., 2009) cannot properly capture phrase patterns. Consider the following example: *not so helpful* vs. *not so bad*. In a unigram-based regression model each unigram gets a weight indicating its polarity and strength. High positive/negative weights are strongly positive/negative clues. It is reasonable to assign a positive weight to *helpful* and a negative weight to *bad*. The fundamental problem of unigrams arises when assigning a weight to *not*. If *not* had a strongly negative weight, the positive weight of *helpful* would be strongly reduced while the negative weight of *bad* would be amplified (by combining weights). This clearly fails to capture the true intentions of the opinion phrases. The same problem holds for *so*, which is an intensifier that should keep the same sign as the word it modifies. We refer to this limitation of the unigram model as **polarity incoherence**.

A promising way of overcoming this weakness is to include *n-grams*, generalizing the bag-of-words model into a bag-of-phrases model (Baccianella et al., 2009; Pang and Lee, 2008). However, regression models over the feature space of all n-grams (for either fixed maximal n or variable-length phrases) are computationally expensive in their training phase. Moreover and most importantly for our setting, including n-grams in the model results in a very high dimensional feature space: many features will then occur only very rarely in the training data. Therefore, it is difficult if not impossible to reliably learn n-gram weights from limited-size training sets. We refer to this problem as the **n-gram sparsity bottleneck**. In our experiments we investigate the effect of using bigrams and variable-length ngrams for improving review rating prediction.

1.2 Contribution

To overcome the above limitations of unigram and n-gram features, we have developed a novel kind of *bag-of-opinions* model, which exploits domain-independent corpora of opinions (e.g., all Amazon reviews), but is finally applied for learning predictors on domain-specific reviews (e.g., movies as rated in IMDB or Rottentomatoes). A document is represented as a bag of opinions each of which has three components: a root word, a set of modifier words and one or more negation words. In the phrase *not very helpful*, the opinion root is *helpful*, one (of potentially many) opinion modifier(s) is *very*, and a negation word is *not*. We enforce polarity coherence by the design of a learnable function that assigns a score to an opinion.

Our approach generalizes the cumulative linear offset model (CLO) presented in (Liu and Seneff, 2009). The CLO model makes several restrictive assumptions, most notably, that all opinion scores within one document are the same as the overall document rating. This assumption does not hold in practice, not even in reviews with extremely positive/negative ratings. For example, in a 5-star Amazon review the phrases *most impressive book* and *it helps explain* should receive different scores. Otherwise, the later transfer step to different domains would yield poor predictions. Due to this restriction, CLO works well on particular types of reviews that have pro/con entries listing characteristic major opinions about the object under review. For settings with individual reviews whose texts do not exhibit any specific structure, the CLO model faces its limitations.

In our bag-of-opinions method, we address the learning of opinion scores as a constrained ridge regression problem. We consider the opinion scores in a given review to be drawn from an unknown probability distribution (so they do not have to be the same within a document). We estimate the review rating based on a set of statis-

tics (e.g., expectation, variance, etc.) derived from the scores of opinions in a document. Thus, our method has a sound statistical foundation and can be applied to arbitrary reviews with mixed opinion polarities and strengths. We avoid the n-gram sparsity problem by the limited-size structured feature space of *(root,modifiers,negators)* opinions.

We treat domain-independent and domain-dependent opinions differently in our system. In the first step we learn a bag-of-opinions model on a large dataset of online reviews to obtain scores for domain-independent opinions. Since the polarity of opinions is not bound to a topic, one can learn opinion scores from a pooled corpus of reviews for various categories, e.g., movies, books, etc., and then use these scored opinions for predicting the ratings of reviews belonging to a particular category. In order to also capture domain-dependent information (possibly complementary to the opinion lexicon used for learning domain-independent opinions), we combine the bag-of-opinions model with an unigram model trained on the domain-dependent corpus. Since domain-dependent training is typically limited, we model it using unigram models rather than bag-of-opinions. By combining the two models, even if an opinion does not occur in the domain-dependent training set but it occurs in a test review, we can still accurately predict the review rating based on the globally learned opinion score. In some sense our combined learning scheme is similar to smoothing in standard learning techniques, where the estimate based on a limited training set is smoothed using a large background corpus (Zhai and Lafferty, 2004).

In summary, the contributions of this paper are the following:

1. We introduce the bag-of-opinions model, for capturing the influence of n-grams, but in a structured way with root words, modifiers, and negators, to avoid the explosion of the feature space caused by explicit n-gram models.

2. We develop a constrained ridge regression method for learning scores of opinions from domain-independent corpora of rated reviews.

3. For transferring the regression model to newly given domain-dependent applications, we derive a set of statistics over opinion scores in documents and use these as features, together with standard unigrams, for predicting the rating of a review.

4. Our experiments with Amazon reviews from different categories (books, movies, music) show that the bag-of-opinions method outperforms prior state-of-the-art techniques.

2 Bag-of-Opinions Model

In this section we first introduce the bag-of-opinions model, followed by the method for learning (domain-independent) model parameters. Then we show how we annotate opinions and how we adapt the model to domain-dependent data.

2.1 Model Representation

We model each document as a bag-of-opinions $\{op_k\}_{k=1}^{K}$, where the number of opinions K varies among documents. Each opinion op_k consists of an opinion root w_r, $r \in S_R$, a set of opinion modifiers $\{w_m\}_{m=1}^{M}$, $m \in S_M$ and a set of negation words $\{w_z\}_{z=1}^{Z}$, $z \in S_Z$, where the sets S_R, S_M, S_Z are component index sets of opinion roots, opinion modifiers and negation words respectively. The union of these sets forms a global component index set $S \in \mathbb{N}^d$, where d is the dimension of the index space. The opinion root determines the prior polarity of the opinion. Modifiers intensify or weaken the strength of the prior polarity. Negation words strongly reduce or reverse the prior polarity. For each opinion, the set of negation words consists of at most a negation valence shifter like *not* (Kennedy and Inkpen, 2006) and its intensifiers like capitalization of the valence shifter. Each opinion component is associated with a score. We assemble the scores of opinion elements into an opinion-score by using a score function. For example, in the opinion *not very helpful*, the opinion root *helpful* determines the prior polarity positive say with a score 0.9, the modifier *very* intensifies the polarity say with a

score 0.5. The prior polarity is further strongly reduced by the negation word *not* with e.g., a score -1.2. Then we sum up the scores to get a score of 0.2 for the opinion *not very helpful*.

Formally, we define the function $score(op)$ as a linear function of opinion components, which takes the form

$$\begin{aligned} score(op) &= sign(r)\beta_r x_r \\ &+ \sum_{m=1}^{M} sign(r)\beta_m x_m \\ &+ \sum_{z=1}^{Z} sign(r)\beta_z x_z \end{aligned} \quad (1)$$

where $\{x_z, x_m, x_r\}$ are binary variables denoting the presence or absence of negation words, modifiers and opinion root. $\{\beta_z, \beta_m, \beta_r\}$ are weights of each opinion elements. $sign(r) : w_r \rightarrow \{-1, 1\}$ is the opinion polarity function of the opinion root w_r. It assigns a value 1/-1 if an opinion root is positive/negative. Due to the semantics of opinion elements, we have constraints that $\beta_r \geq 0$ and $\beta_z \leq 0$. The sign of β_m is determined in the learning phase, since we have no prior knowledge whether it intensifies or weakens the prior polarity.

Since a document is modeled as a bag-of-opinions, we can simply consider the expectation of opinion scores as the document rating. If we assume the scores are uniformly distributed, the prediction function is then $f(\mathbf{x}) = \frac{1}{K}\sum_{k=1}^{K} score(op_k)$ which assigns the average of opinion scores to the document $\mathbf{x}$.

2.2 Learning Regression Parameters

We assume that we can identify the opinion roots and negation words from a subjectivity lexicon. In this work we use MPQA (Wilson et al., 2005). In addition, the lexicon provides the prior polarity of the opinion roots. In the training phase, we are given a set of documents with ratings $\{\mathbf{x}_i, y_i\}_{i=1}^{N}$, and our goal is to find an optimal function f^* whose predictions $\{\hat{y}_i\}_{i=1}^{N}$ are as close as possibile to the original ratings $\{y_i\}_{i=1}^{N}$. Formally, we aim to minimize the following loss function:

$$L = \frac{1}{2N}\sum_{i=1}^{N}(f(\mathbf{x}_i) - y_i)^2 \quad (2)$$

where $f(\mathbf{x}_i)$ is modeled as the average score of opinions in review $\mathbf{x}_i$.

First, we rewrite $score(op)$ as the dot product $\langle \boldsymbol{\beta}, \mathbf{p} \rangle$ between a weight vector $\boldsymbol{\beta} = [\boldsymbol{\beta}_z, \boldsymbol{\beta}_m, \beta_r]$ and a feature vector $\mathbf{p} = [sign(r)\mathbf{x}_z, sign(r)\mathbf{x}_m, sign(r)x_r]$. In order to normalize the vectors, we rewrite the weight and feature vectors in the d dimensional vector space of all root words, modifiers and negation words. Then $\boldsymbol{\beta} = [.., \boldsymbol{\beta}_z, 0, .., \boldsymbol{\beta}_m, 0, .., \beta_r, 0..] \in R^d$ and $\mathbf{p} = [sign(r)\mathbf{x}_z, 0, .., sign(r)\mathbf{x}_m, 0, .., sign(r)x_r, ...] \in R^d$. The function $f(\mathbf{x_i})$ can then be written as the dot product $\langle \boldsymbol{\beta}, \mathbf{v_i} \rangle$, where $\mathbf{v_i} = \frac{1}{K_i}\sum_{k=1}^{K_i} \mathbf{p}_k$, with K_i the number of opinions in review $\mathbf{x_i}$. By using this feature representation, the learning problem is equivalent to:

$$\min_{\beta} \; L(\boldsymbol{\beta}) = \frac{1}{2N}\sum_{i=1}^{N}(\langle \boldsymbol{\beta}, \mathbf{v}_i \rangle + \beta_0 - y_i)^2$$

$s.t.$

$$\begin{aligned} \beta_z &\leq 0 \quad z \in S_Z \\ \beta_r &\geq 0 \quad r \in S_R \end{aligned} \quad (3)$$

where $\boldsymbol{\beta} \in R^d$, $\boldsymbol{\beta} = [\boldsymbol{\beta}_z, \boldsymbol{\beta}_m, \boldsymbol{\beta}_r]$. β_0 is the intercept of the regression function, which is estimated as the mean of the ratings in the training set. We define a new variable $\tilde{y}_i = y_i - \beta_0$.

In order to avoid overfitting, we add an $l2$ norm regularizer to the loss function with the parameter $\lambda > 0$.

$$LR(\boldsymbol{\beta}) = \frac{1}{2N}\sum_{i=1}^{N}(\langle \boldsymbol{\beta}, \mathbf{v}_i \rangle - \tilde{y}_i)^2 + \frac{\lambda}{2} \parallel \boldsymbol{\beta} \parallel_2^2$$

$s.t.$

$$\begin{aligned} \beta_z &\leq 0 \quad z \in S_Z \\ \beta_r &\geq 0 \quad r \in S_R \end{aligned} \quad (4)$$

We solve the above optimization problem by Algorithm 1 using coordinate descent. The procedure starts with $\boldsymbol{\beta}^0 = 0$, $\boldsymbol{\beta}^0 \in R^d$. Then it updates iteratively every coordinate of the vector $\boldsymbol{\beta}$ until convergence. Algorithm 1 updates every coordinate $\beta_j, j \in \{1, 2, ..., d\}$ of $\boldsymbol{\beta}$ by solving the following one-variable sub-problem:

$$min_{l_j \leq \beta_j \leq c_j} LR(\beta_1, ..., \beta_j, ..., \beta_d)$$

where l_j and c_j denote the lower and upper bounds of β_j. If $j \in S_Z$, $l_j = -\infty$ and $c_j = 0$. If $j \in S_R$, $l_j = 0$ and $c_j = \infty$. Otherwise both bounds are infinity.

According to (Luo and Tseng, 1992), the solution of this one-variable sub-problem is

$$\hat{\beta}_j = max\{l_j, min\{c_j, g_j\}\}$$

where

$$g_j = \frac{\frac{1}{N}\sum_{i=1}^{N} v_{ij}(\tilde{y}_i - \sum_{l \neq j} \beta_l v_l)}{\frac{1}{N}\sum_{i=1}^{N} v_{ij}^2 + \lambda}$$

Here g_j is the close form solution of standard ridge regression at coordinate j (for details see (Friedman et al., 2008)). We prove the convergence of Algorithm 1, by the following theorem using techniques in (Luo and Tseng, 1992).

Theorem 1 *A sequence of $\boldsymbol{\beta}$ generated by Algorithm 1 globally converges to an optimal solution $\beta^* \in \chi^*$ of problem (4), where χ^* is the set of optimal solutions.*

Proof: *Luo and Tseng (1992) show that coordinate descent for constrained quadratic functions in the following form converges to one of its global optimal solutions.*

$$\begin{aligned} min_{\boldsymbol{\beta}} \quad & h(\boldsymbol{\beta}) = \langle \boldsymbol{\beta}, \mathbf{Q}\boldsymbol{\beta} \rangle / 2 + \langle \mathbf{q}, \boldsymbol{\beta} \rangle \\ s.t. \quad & \mathbf{E}^T \boldsymbol{\beta} \geq \mathbf{b} \end{aligned}$$

where $\mathbf{Q}$ is a $d \times d$ symmetric positive-definite matrix, $\mathbf{E}$ is a $d \times d$ matrix having no zero column, $\mathbf{q}$ is a d-vector and $\mathbf{b}$ is a d-vector.

We rewrite LR in matrix form as

$$\begin{aligned} & \frac{1}{2N}(\tilde{\mathbf{y}} - \mathbf{V}\boldsymbol{\beta})^T(\tilde{\mathbf{y}} - \mathbf{V}\boldsymbol{\beta}) + \frac{\lambda}{2}\boldsymbol{\beta}^T\boldsymbol{\beta} \\ = \; & \frac{1}{2N}(\mathbf{V}\boldsymbol{\beta})^T(\mathbf{V}\boldsymbol{\beta}) + \frac{\lambda}{2}\boldsymbol{\beta}^T\boldsymbol{\beta} - \frac{1}{2N}((\mathbf{V}\boldsymbol{\beta})^T\tilde{\mathbf{y}} \\ & - \frac{1}{2N}\tilde{\mathbf{y}}^T(\mathbf{V}\boldsymbol{\beta})) + \frac{1}{2N}\tilde{\mathbf{y}}^T\tilde{\mathbf{y}} \\ = \; & \langle \boldsymbol{\beta}, \mathbf{Q}\boldsymbol{\beta} \rangle / 2 + \langle \mathbf{q}, \boldsymbol{\beta} \rangle + constant \end{aligned}$$

where

$$\mathbf{Q} = \mathbf{B}^T\mathbf{B}, \mathbf{B} = \begin{bmatrix} \sqrt{\frac{1}{N}}\mathbf{V} \\ \sqrt{\lambda}\mathbf{I}^{d \times d} \end{bmatrix}, \mathbf{q} = \frac{-1}{N}(\mathbf{V}^T\tilde{\mathbf{y}})$$

where $\mathbf{I}^{d \times d}$ is the identity matrix. Because $\lambda > 0$, all columns of $\mathbf{B}$ are linearly independent. As $\mathbf{Q} = \mathbf{B}^T\mathbf{B}$ and symmetric, $\mathbf{Q}$ is positive definite.

We define $\mathbf{E}$ as a $d \times d$ diagonal matrix with all entries on the main diagonal equal to 1 except $e_{ii} = -1, i \in S_Z$ and $\mathbf{b}$ is a d-vector with all entries equal to $-\infty$ except $b_i = 0$, for $i \in S_Z$ or $i \in S_R$.

Because the almost cyclic rule is applied to generate the sequence $\{\boldsymbol{\beta}^t\}$, the algorithm converges to a solution $\boldsymbol{\beta}^* \in \chi^*$.

Algorithm 1 Constrained Ridge Regression

1: Input: λ and $\{\mathbf{v}_n, \tilde{y}_n\}_{n=1}^N$
2: Output: optimal $\boldsymbol{\beta}$
3: **repeat**
4: **for** $j = 1, ..., d$ **do**
5: $g_j = \frac{\frac{1}{N}\sum_{i=1}^{N} v_{ij}(\tilde{y}_i - \sum_{l \neq j} \beta_l v_l)}{\frac{1}{N}\sum_{i=1}^{N} v_{ij}^2 + \lambda}$
6:

$$\hat{\beta}_j = \begin{cases} 0, & if\ j \in S_R\ and\ g_j < 0 \\ 0, & if\ j \in S_Z\ and\ g_j > 0 \\ g_j, & else \end{cases}$$

7: **end for**
8: **until** Convergence condition is satisfied

2.3 Annotating Opinions

The MPQA lexicon contains separate lexicons for subjectivity clues, intensifiers and valence shifters (Wilson et al., 2005), which are used for identifying opinion roots, modifiers and negation words. Opinion roots are identified as the positive and negative subjectivity clues in the subjectivity lexicon. In the same manner, intensifiers and valence shifters of the type {negation, shiftneg} are mapped to modifiers and negation words. Other modifier candidates are adverbs, conjunctions and modal verbs around opinion roots. We consider non-words modifiers as well, e.g., punctuations, capitalization and repetition of opinion roots. If the opinion root is a noun, adjectives are also included into modifier sets.

The automatic opinion annotation starts with locating the continous subjectivity clue sequence. Once we find such a sequence and at least one of the subjectivity clue is positive or negative, we search to the left up to 4 words for negation words and modifier candidates, and stop if encountering another opinion root. Similarly, we search to the

right up to 3 unigrams for modifiers and stop if we find negation words or any other opinion roots. The prior polarity of the subjectivity sequence is determined by the polarity of the last subjectivity clue with either positive or negative polarity in the sequence. The other subjectivity clues in the same sequence are treated as modifiers.

2.4 Adaptation to Domain-Dependent Data

The adaptation of the learned (domain-independent) opinion scores to the target domain and the integration of domain-dependent unigrams is done in a second ridge-regression task. Note that this is a simpler problem than typical domain-adaptation, since we already know from the sentiment lexicon which are the domain-independent features. Additionally, its relatively easy to obtain a large mixed-domain corpus for reliable estimation of domain-independent opinion scores (e.g., use all Amazon product reviews). Furthermore, we need a domain-adaptation step since domain-dependent and domain-independent data have generally different rating distributions. The differences are mainly reflected in the intercept of the regression function (estimated as the mean of the ratings). This means that we need to scale the positive/negative mean of the opinion scores differently before using it for prediction on domain-dependent reviews. Moreover, other statistics further characterize the opinion score distribution. We use the variance of opinion scores to capture the reliability of the mean, multiplied by the negative sign of the mean to show how much it strengthens/weakens the estimation of the mean. The mean score of the dominant polarity (*major exp*) is also used to reduce the influence of outliers. Because positive and negative means should be scaled differently, we represent positive and negative values of the mean and *major exp* as 4 different features. Together with variance, they are the 5 statistics of the opinion score distribution. The second learning step on opinion score statistics and domain-dependent unigrams as features, re-weights the importance of domain-independent and domain-dependent information according to the target domain bias.

3 Experimental Setup

We performed experiments on three target domains of Amazon reviews: books, movies (DVDs), and music (CDs). For each domain, we use ca. 8000 Amazon reviews for evaluation; an additional set of ca. 4000 reviews are withheld for parameter tuning (regularization parameter, etc.). For learning weights for domain-independent opinions, we use a mixed-domain corpus of ca. 350,000 reviews from Amazon (electronics, books, dvds, etc.); this data is disjoint from the test sets and contains no reviews from the music domain. In order to learn unbiased scores, we select about the same number of positive and negative reviews (where reviews with more/less than 3 stars are regarded as positive/negative). The regularization parameters used for this corpus are tuned on withheld data with ca. 6000 thematically mixed reviews.[1].

We compare our method, subsequently referred to as *CRR-BoO* (Constrained Ridge Regression for Bag-of-Opinions), to a number of alternative state-of-the-art methods. These competitors are varied along two dimensions: 1) feature space, and 2) training set. Along the first dimension, we consider a) unigrams coined *uni*, b) unigrams and bigrams together, coined *uni+bi*, c) variable-length n-grams coined *n-gram*, d) the opinion model by (Liu and Seneff, 2009) coined *CLO* (cumulative linear offset model). As learning procedure, we use ridge regression for a), b), and d), and bounded cyclic regression, coined *BCR*, for c). Along the second - orthogonal - dimension, we consider 3 different training sets: i) domain-dependent training set coined *DD*, ii) the large mixed-domain training set coined *MD*, iii) domain-dependent training set and the large mixed-domain training set coined *DD+MD*. For the *DD+MD* training set, we apply our two stage approach for *CRR-BoO* and *CLO*, i.e., we use the mixed-domain corpus for learning the opinion scores in the first stage, and integrate unigrams from *DD* in a second domain-adaptation stage. We train the remaining feature models directly on the combination of the whole mixed-domain cor-

[1] All datasets are available from `http://www.mpi-inf.mpg.de/~lqu`

feature models		uni	uni+bi	n-gram	CLO	**CRR-BoO**
DD	book	1.004	0.961	0.997	1.469	0.942
	dvd	1.062	1.018	1.054	1.554	0.946
	music	0.686	0.672	0.683	0.870	0.638
MD	book	1.696	1.446	1.643	1.714	1.427
	dvd	1.919	1.703	1.858	1.890	1.565
	music	2.395	2.160	2.340	2.301	1.731
DD+MD	book	1.649	1.403	1.611	1.032	**0.884**
	dvd	1.592	1.389	1.533	1.086	**0.928**
	music	1.471	1.281	1.398	0.698	**0.627**

Table 1: Mean squared error for rating prediction methods on Amazon reviews.

pus and the training part of *DD*.

The CLO model is adapted as follows. Since bags-of-opinions generalize CLO, adjectives and adverbs are mapped to opinion roots and modifiers, respectively; negation words are treated the same as CLO. Subsequently we use our regression technique. As Amazon reviews do not contain pro and con entries, we learn from the entire review.

For BCR, we adapt the variable-length n-grams method of (Ifrim et al., 2008) to elastic-net-regression (Friedman et al., 2008) in order to obtain a fast regularized regression algorithm for variable-length n-grams. We search for significant n-grams by incremental expansion in backward direction (e.g., expand *bad* to *not bad*). BCR pursues a dense solution for unigrams and a sparse solution for n-grams. Further details on the BCR learning algorithm will be found on a subsequent technical report.

As for the regression techniques, we show only results with ridge regression (for all feature and training options except BCR). It outperformed ϵ-*support vector regression* (SVR) of libsvm (Chang and Lin, 2001), lasso (Tibshirani, 1996), and elastic net (Zou and Hastie, 2005) in our experiments.

4 Results and Discussion

Table 1 shows the mean square error (MSE) from each of the three domain-specific test sets. The error is defined as $MSE = \frac{1}{N}\sum_{i=1}^{N}(f(\mathbf{x}_i) - y_i)^2$. The right most two columns of the table show results for the full-fledge two-stage learning for our method and CLO, with domain-dependent weight learning and the domain adaptation step. The other models are trained directly on the given training sets. For the DD and DD+MD training sets, we use five-fold cross-validation on the domain-specific sets. For the MD training set, we take the domain-specific test sets as hold-out data for evaluation.

Table 1 clearly shows that our *CRR-BoO* method outperforms all alternative methods by a significant margin. Most noteworthy is the music domain, which is not covered by the mixed-domain corpus. As expected, unigrams only perform poorly, and adding bigrams leads only to marginal improvements. BCR pursues a dense solution for unigrams and a sparse solution for variable-length n-grams, but due to the sparsity of occurence of long n-grams, it filters out many interesting-but-infrequent ngrams and therefore performs worse than the dense solution of the *uni+bi* model. The CLO method of (Liu and Seneff, 2009) shows unexpectedly poor performance. Its main limitation is the assumption that opinion scores are identical within one document. This does not hold in documents with mixed opinion polarities. It also results in conflicts for opinion components that occur in both positive and negative documents. In contrast, *CRR-BoO* naturally captures the mixture of opinions as a bag of positive/negative scores. We only require that the mean of opinion scores equals the overall document rating.

The right most column of Table 1 shows that our method can be improved by learning opinion scores from the large mixed-domain corpus. How-

opinion	score
good	0.18
recommend	1.64
most difficult	-1.66
but it gets very good!	2.37
would highly recommend	2.73
would not recommend	-1.93

Table 2: Example opinions learned from the Amazon mixed-domain corpus.

ever, the high error rates of the models learned directly on the MD corpus show that direct training on the mixed-domain data can introduce a significant amount of noise into the prediction models. Although the noise can be reduced by learning from MD and DD together, the performance is still worse than when learning directly from the domain-dependent corpora. Additionally, when the domain is not covered by the mixed-domain corpus (e.g., music), the results are even worse. Thus, the two stages of our method (learning domain-independent opinion scores plus domain-adaptation) are decisive for a good performance, and the sentiment-lexicon-based BoO model leads to robust learning of domain-independent opinion scores.

Another useful property of BoO is its high interpretability. Table 2 shows example opinion scores learned from the mixed-domain corpus. We observe that the scores corelate well with our intuitive interpretation of opinions.

Our *CRR-BoO* method is highly scalable. Excluding the preprocessing steps (same for all methods), the learning of opinion component weights from the ca. 350,000 domain-independent reviews takes only 11 seconds.

5 Related Work

Rating prediction is modeled as an ordinal regression problem in (Pang and Lee, 2005; Goldberg and Zhu, 2006; Snyder and Barzilay, 2007). They simply use the bag-of-words model with regression algorithms, but as seen previously this cannot capture the expressive power of phrases. The resulting models are not highly interpretable. Baccianella et al. (2009) restrict the n-grams to the ones having certain POS patterns. However, the long n-grams matching the patterns still suffer from sparsity. The same seems to hold for sparse n-gram models (BCR in this paper) in the spirit of Ifrim et al. (2008). Although sparse n-gram models can explore arbitrarily large n-gram feature spaces, they can be of little help if the n-grams of interests occur sparsely in the datasets.

Since our approach can be regarded as learning a domain-independent sentiment lexicon, it is related to the area of automatically building domain-independent sentiment lexicons (Esuli and Sebastiani, 2006; Godbole et al., 2007; Kim and Hovy, 2004). However, this prior work focused mainly on the opinion polarity of opinion words, neglecting the opinion strength. Recently, the lexicon based approaches were extended to learn domain-dependent lexicons (Kanayama and Nasukawa, 2006; Qiu et al., 2009), but these approaches also neglect the aspect of opinion strength. Our method requires only the prior polarity of opinion roots and can thus be used on top of those methods for learning the scores of domain-dependent opinion components. The methods proposed in (Hu and Liu, 2004; Popescu and Etzioni, 2005b) can also be categorized into the lexicon based framework because their procedure starts with a set of seed words whose polarities are propagated to other opinion bearing words.

6 Conclusion and Future Work

In this paper we show that the bag-of-opinions (BoO) representation is better suited for capturing the expressive power of n-grams while at the same time overcoming their sparsity bottleneck. Although in this paper we use the BoO representation to model domain-independent opinions, we believe the same framework can be extended to domain-dependent opinions and other NLP applications which can benefit from modelling n-grams (given that the n-grams are decomposable in some way). Moreover, the learned model can be regarded as a domain-independent opinion lexicon with each entry in the lexicon having an associated score indicating its polarity and strength. This in turn has potential applications in sentiment summarization, opinionated information retrieval and opinion extraction.

References

Baccianella, S., A. Esuli, and F. Sebastiani. 2009. Multi-facet rating of product reviews. In *ECIR*. Springer.

Chang, C.C. and C.J. Lin, 2001. *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Esuli, A. and F. Sebastiani. 2006. Sentiwordnet: A publicly available lexical resource for opinion mining. In *LREC*, pages 417–422.

Friedman, J., T. Hastie, and R. Tibshirani. 2008. Regularization paths for generalized linear models via coordinate descent. Technical report, Technical Report, Available at http://www-stat. stanford. edu/jhf/ftp/glmnet. pdf.

Ganu, G., N. Elhadad, and A. Marian. 2009. Beyond the stars: Improving rating predictions using review text content. In 12^{th} *International Workshop on the Web and Databases*.

Godbole, Namrata, Manjunath Srinivasaiah, and Steven Skiena. 2007. Large-scale sentiment analysis for news and blogs. In *ICWSM*.

Goldberg, A. B. and X.J. Zhu. 2006. Seeing stars when there aren't many stars: Graph-based semi-supervised learning for sentiment categorization. In *HLT-NAACL 2006 Workshop on Textgraphs: Graph-based Algorithms for Natural Language Processing*.

Hu, M.Q. and B. Liu. 2004. Mining and summarizing customer reviews. In *CIKM*, pages 168–177. ACM New York,USA.

Ifrim, G., G. Bakir, and G. Weikum. 2008. Fast logistic regression for text categorization with variable-length n-grams. In *KDD*, pages 354–362, New York,USA. ACM.

Kanayama, H. and T. Nasukawa. 2006. Fully automatic lexicon expansion for domain-oriented sentiment analysis. In *EMNLP*, pages 355–363.

Kennedy, A. and D. Inkpen. 2006. Sentiment classification of movie reviews using contextual valence shifters. *Computational Intelligence*, 22(2):110–125.

Kim, S.M. and E. Hovy. 2004. Determining the sentiment of opinions. In *COLING*, pages 1367–1373.

Lerman, K., S. Blair-Goldensohn, and R. McDonald. 2009. Sentiment summarization: Evaluating and learning user preferences. In *EACL*, pages 514–522. ACL.

Liu, J.J. and S. Seneff. 2009. Review Sentiment Scoring via a Parse-and-Paraphrase Paradigm. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 161–169. ACL.

Luo, Z.Q. and Q. Tseng. 1992. On the convergence of the coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, 72(1):7–35.

Pang, B. and L. Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*, page 124. ACL.

Pang, B. and L. Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

Popescu, A.M. and O. Etzioni. 2005a. Extracting product features and opinions from reviews. In *HLT/EMNLP*, volume 5, pages 339–346. Springer.

Popescu, A.M. and O. Etzioni. 2005b. Extracting product features and opinions from reviews. In *Proceedings of HLT/EMNLP*, volume 5, pages 339–346. Springer.

Qiu, G., B. Liu, J.J. Bu, and C. Chen. 2009. Expanding Domain Sentiment Lexicon through Double Propagation. In *IJCAI*.

Snyder, B. and R. Barzilay. 2007. Multiple aspect ranking using the good grief algorithm. In *NAACL/HLT*, pages 300–307.

Tibshirani, R. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288.

Titov, I. and R. McDonald. 2008a. A joint model of text and aspect ratings for sentiment summarization. In *HLT/ACL*, pages 308–316.

Titov, I. and R. McDonald. 2008b. Modeling online reviews with multi-grain topic models. In *WWW*, pages 111–120. ACM.

Wilson, T., J. Wiebe, and P. Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *HLT/ACL*, pages 347–354.

Zhai, C. X. and J. Lafferty. 2004. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, 22(2):179–214.

Zou, H. and T. Hastie. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B(Statistical Methodology)*, 67(2):301–320.

An Exploration of Features for Recognizing Word Emotion

Changqin Quan
Faculty of Engineering
University of Tokushima
quan-c@is.tokushima-u.ac.jp

Fuji Ren
Faculty of Engineering
University of Tokushima
ren@is.tokushima-u.ac.jp

Abstract

Emotion words have been well used as the most obvious choice as feature in the task of textual emotion recognition and automatic emotion lexicon construction. In this work, we explore features for recognizing word emotion. Based on RenCECps (an annotated emotion corpus) and MaxEnt (Maximum entropy) model, several contextual features and their combination have been experimented. Then PLSA (probabilistic latent semantic analysis) is used to get semantic feature by clustering words and sentences. The experimental results demonstrate the effectiveness of using semantic feature for word emotion recognition. After that, "word emotion components" is proposed to describe the combined basic emotions in a word. A significant performance improvement over contextual and semantic features was observed after adding word emotion components as feature.

1 Introduction

Textual emotion analysis is becoming increasingly important due to augmented communication via computer mediated communication (CMC). A possible application of textual emotion recognition is online chat system. An emotion feedback system can recognize users' emotion and give appropriate responses. Another application example is weblog emotion recognition and prediction. Blogspace consists of millions of users who maintain their online diaries, containing frequently-updated views and personal remarks about a range of issues. An emotion recognition and prediction system can understand the public's reaction to some social issues and predict emotion changes. It would be helpful for solving some psychological problems or giving early warnings, such as suicide or terrorism.

Textual emotion analysis also can improve the accuracy of other nonverbal modalities like speech or facial emotion recognition, and to improve human computer interaction systems. However, automatic recognition of emotion meaning from texts presents a great challenge. One of the reasons is the manifoldness of expressed emotions in words.

Emotion words have been well used as the most obvious choice as feature in the task of textual emotion recognition and automatic emotion lexicon construction (Virginia and Pablo, 2006; Tokuhisa et al., 2008, etc.). And there are many lexical resources developed for these tasks, such as GI (Stone et al., 1966), WordNet-Affect (Strapparava and Valitutti, 2004), NTU Sentiment Dictionary (Ku et al., 2006), Hownet (Dong and Dong, 2003), SentiWordnet (Esuli and Sebastiani, 2006). In these sentimental or affective lexicons, the words usually bear direct emotions or opinions, such as happy or sad, good or bad. Although they play a role in some applications, several problems of emotion expression in words have been ignored.

Firstly, there are a lot of sentences can evoke emotions without direct emotion words. For example,

(1) 春天在孩子们的眼里、在孩子们的心里。(Spring is in children's eyes, and in their hearts.)

In sentence (1), we may feel joy, love or expect delivered by the writer. But there are no direct emotion words can be found from lexicons. As Ortony (1987) indicates, besides words directly referring to emotion states (e.g., "fear", "cheerful") and for which an appropriate lexicon would help, there are words that act only as an indirect

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 922–930,
Beijing, August 2010

reference to emotions depending on the context. Strapparava et al. (2006) also address this issue. The authors believed that all words can potentially convey affective meaning, and they distinguished between words directly referring to emotion states (direct affective words) and those having only an indirect reference that depends on the context (indirect affective words).

The second problem is emotion ambiguity of words. The same word in different contexts may reflect different emotions. For example,

(2) 这是目前我**唯**一能做的。(This is currently the only thing I can do.)

(3) 他是我的**唯**一。(He is my only one.)

In sentence (2), the word "唯一 (only)" may express the emotion of anxiety or expect; but in sentence (3), the word "唯一 (only)" may express the emotion of love or expect. The emotion categories can not be determined without their certain contexts especially for the words with emotion ambiguity.

In addition, some words can express multiple emotions, such as "悲喜交加 (mingled feelings of joy and sorrow)". Statistics on an annotated emotion corpus (Ren-CECps [1], Chinese emotion corpus developed by Ren-lab) showed that 84.9% of all emotion words have one emotion, 15.1% have more than one emotions (Quan and Ren, 2010). Multi-emotion words are indispensable for expressing complex feelings in use of language.

In this work, we explore features for recognizing word emotion in sentences. Based on Ren-CECps and MaxEnt model, several contextual features and their combination have been experimented. Then PLSA (probabilistic latent semantic analysis) is used to get semantic feature by clustering word and sentence. The experimental results demonstrate the effectiveness of using semantic feature for word emotion recognition. After that, the notion of "word emotion components" is proposed to describe the combined basic emotions in a word. A significant performance improvement over only using contextual and semantic features was observed after adding word emotion components as feature and output in MaxEnt based model.

[1] http://a1-www.is.tokushima-u.ac.jp/member/ren/Ren-CECps1.0/Ren-CECps1.0.html

This paper is organized as follows. In section 2, based on Ren-CECps and MaxEnt, an exploration of using contextual feature for Chinese word emotion recognition is described. In section 3, using PLSA technique, the performance of adding semantic feature is presented. In section 4, the notion of "word emotion components" is proposed and the performance of using encoding feature is presented. In section 5, the discussions are described. Section 6 is conclusions.

2 Chinese Word Emotion Recognition

2.1 Related Works

There are many researches concerning computing semantics of words, while the researches on computing emotions of words are relatively less. Computing word emotions is a challenge task because the inherent of emotion is ambiguous and natural language is very rich in emotion terminology. Using the textual emotion information, several methods have been explored for computing lexical emotions. Wilson et al. (2009) proposed a two-step approach to classify word polarity out of context firstly, and then to classify word polarity in context with a wide variety of features. Strapparava et al. (2007) implemented a variation of Latent Semantic Analysis (LSA) to measure the similarities between direct affective terms and generic terms. Lee and Narayanan (2005) proposed a method of computing mutual information between a specific word and emotion category to measure how much information a word provides about a given emotion category (emotion salience). Based on structural similarity, Bhowmick (2008) computed the structural similarity of words in WordNet to distinguish the emotion words from the non-emotion words. Kazemzadeh (2008) measured similarity between word and emotion category based on interval type-2 fuzzy logic method. Takamura (2005) used a spin model to extract emotion polarity of words.

Different from the above researches, in this work, we explore which features are effective for word emotion recognition. The features include contextual feature, semantic feature and encoding feature.

2.2 Ren-CECps and MaxEnt based Chinese Word Emotion Recognition

Ren-CECps is constructed based on a relative fine-grained annotation scheme, annotating emotion in text at three levels: document, paragraph, and sentence. The all dataset consisted of 1,487 blog articles published at sina blog, sciencenet blog, etc. There are 11,255 paragraphs, 35,096 sentences, and 878,164 Chinese words contained in this corpus (more details can be found in (Quan and Ren, 2010)).

In the emotion word annotation scheme of Ren-CECps, direct emotion words and indirect emotion words in a sentence are all annotated. For example, in sentence (1) "春天 (spring)" and "孩子们 (the children)" are labeled. An emotion keyword or phrase is represented as a vector to record its intensities of the eight basic emotion classes (expect, joy, love, surprise, anxiety, sorrow, angry and hate). For instance, the emotion vector for the word "春天 (spring)" $\vec{w} = (0.1, 0.3, 0.3, 0.0, 0.0, 0.0, 0.0, 0.0)$ indicates the emotions of weak expect, joy and love. In this work, we focus on if a word contains some emotion(s) in a certain context. The analysis on emotion intensity of emotion words is included in our future work.

As word emotion is subjective entity, a word in a certain context may evoke multiple emotions in different people's mind. A part of documents in Ren-CECps have been annotated by three annotators independently to measure agreement on the annotation of this corpus, which include 26 documents with a total of 805 sentences, 19,738 words. This part of corpus is used as testing corpus to evaluate the experimental results. (Section 5.1 shows the analysis on the annotation agreement on word emotion.)

MaxEnt modeling provides a framework for integrating information from many heterogeneous information sources for classification (Manning, 1999). MaxEnt principle is a well used technique provides probability of belongingness of a token to a class. In word emotion recognition, the MaxEnt estimation process produces a model in which each feature f_i is assigned a weight α_i. The deterministic model produces conditional probability (Berger, 1996), see equation (1) and (2). In experiments, we have used a Java based open-nlp MaxEnt toolkit [2].

$$p(e|context) = \frac{1}{Z(context)} \prod_i \alpha_i^{f_i(context,e)} \quad (1)$$

$$Z(context) = \sum \prod_i \alpha_i^{f_i(context,e)} \quad (2)$$

2.3 Contextual Features

The contextual features used in MaxEnt for Chinese word emotion recognition are described as follows:

Word Feature (WF): Word itself to be recognized.

N-words Feature (NF): To know the relationship between word emotion and its context, the surrounding words of length n for the word (w_i) to be recognized are used as feature: ($w_{i-n}...w_i...w_{i+n}$).

POS Feature (POSF): The part of speech of the current word and surrounding words are used as feature. We have used a Chinese segmentation and POS tagger (Ren-CMAS) developed by Ren-lab, which has an accuracy about 97%. The set of POS includes 35 classes.

Pre-N-words Emotion Feature (PNEF): The emotions of the current word may be influenced by the emotions of its previous words. So the emotions of previous n words are used as feature. The value of this feature for a word (w_i) is obtained only after the computation of the emotions for its previous words.

Pre-is-degree-word Feature (PDF), Pre-is-negative-word Feature (PNF), Pre-is-conjunction Feature (PCF): To determine if the previous word is a degree word, a negative word, or a conjunction may be helpful to identify word emotions. The degree word list (contains 1,039 words), negative word list (contains 645 words), and conjunction list (contains 297 words) extracted from Ren-CECps have been used.

2.4 The Performance of Using Contextual Feature

We use the documents in Ren-CECps that have been annotated by three annotators independently

[2] http://maxent.sourceforge.net/

as testing corpus. An output of word emotion(s) will be regarded as a correct result if it is in agreement with any one item of word emotion(s) provided by the three annotators. The numbers of training and testing corpus are shown in table 1. The accuracies are measured by F-value.

Table 1: Number of training and testing corpus

Number	Training	Testing
Documents	1,450	26
Sentences	33,825	805
Words	813,507	19,738
Emotion words	99,571	2,271*

(*) At least agreed by two annotators.

Table 2 gives the results of F-value for different contextual features in the MaxEnt based Chinese word emotion recognition. The results of F-value include: (a) recognize emotion and unemotion words; (b) recognize the eight basic emotions for emotion words (complete matching); (c) recognize the eight basic emotions for emotion words (single emotion matching).

As shown in table 2, when we only use Word Feature(WF), the F-value of task (a) achieved a high value (96.3). However, the F-values of task (b) and (c) are relative low, that means the problem of recognizing the eight basic emotions for emotion words is a lot more difficult than the problem of recognizing emotion and unemotion words, so we focus on task (b) and (c).

When we experiment with Word Feature(WF) and N-words Feature (NF), we have observed that word feature (w_i) and a window of previous and next word (w_{i-1}, w_i, w_{i+1}) give the best results (a=96.5, b=50.4, c=69.0). Compared with (w_{i-1}, w_i, w_{i+1}), a larger window of previous and next two words ($w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}$) reduces the F-value. This demonstrates that w_i and w_{i-1}, w_i, w_{i+1} are effective features for word emotion recognition.

When POS Feature (POSF) is added, the F-value is increased. Especially the F-value is increased to (a=97.1, b=51.9, c=72.0) when pos_i and $pos_{i-1}, pos_i, pos_{i+1}$ are added.

We also find that Pre-N-words Emotion Feature (PNEF) ($pre\_e_0, ..., pre\_e_{i-1}$) increases the F-value, but previous one word emotion can not increases the F-value.

As can be seen from table 2, when only contextual features are used, the highest F-value is (a=97.1, b=53.0, c=72.7) when Pre-is-degree-word Feature (PDF), Pre-is-negative-word Feature (PNF), Pre-is-conjunction Feature (PCF) are added.

3 Semantic Feature

To know if semantic information is useful for emotion recognition, we have used probabilistic latent semantic analysis (PLSA) (Hofmann, 1999) to cluster words and sentences. PLSA clusters documents based on the term-document co-occurrence which results in semantic decomposition of the term-document matrix into a lower dimensional latent space. PLSA can be defined as:

$$P(s,w) = \sum_{z \in Z} P(z)P(s|z)P(w|z) \qquad (3)$$

where $p(s,w)$ is the probability of word w and sentence s co-occurrence, $P(s|z)$ is the probability of a sentence given a semantic class z, and $P(w|z)$ is the probability of a word given a semantic class z.

For word clustering, We made the assignment based on the maximum $p(z|w)$, if $p(z^{'}|w) = \max p(z|w)$, then w was assigned to $z^{'}$. Sentence clustering is similar to word clustering. Word clustering and sentence clustering are run separately. The word class_id and sentence class_id are used as semantic feature (SF), which including sentence class feature (SCF) and word class feature (WCF). PeenAspect implementation of PLSA has been used for our expriments [3].

Table 3 gives the results of F-value for combined all contextual features and semantic feature in the MaxEnt based Chinese word emotion recognition.

As can be seen from table 3, when SCF is used, the best result is obtained when the cluster number is 100; when WCF is used, the best result is obtained when the cluster number is 100 or 160. The results demonstrate the effectiveness of using SCF is a little higher than using WCF.

[3] http://www.cis.upenn.edu/datamining/software_dist/PennAspect/

Table 2: F-value for different contextual features in the MaxEnt based Chinese word emotion recognition

(a) recognize emotion or unemotion words
(b) recognize the eight basic emotions for emotion words (complete matching)
(c) recognize the eight basic emotions for emotion words (single emotion matching)

Feature type	Features	F-value		
		(a)	(b)	(c)
WF	$f1 = w_i$	96.3	45.9	63.0
NF	$f1 = w_{i-1}, w_i, w_{i+1}$	94.8	44.8	60.7
	$f1 = w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}$	92.4	28.4	40.3
WF+NF	$f1 = w_i$; $f2 = w_{i-1}, w_i, w_{i+1}$	96.5	50.4	69.0
WF+NF +POSF	$f1 = w_i$ $f2 = w_{i-1}, w_i, w_{i+1}$ $f3 = pos_i$	96.8	51.5	71.1
	$f1 = w_i$ $f2 = w_{i-1}, w_i, w_{i+1}$ $f3 = pos_{i-1}, pos_i, pos_{i+1}$	97.0	51.7	71.6
	$f1 = w_i$ $f2 = w_{i-1}, w_i, w_{i+1}$ $f3 = pos_i$ $f4 = pos_{i-1}, pos_i, pos_{i+1}$	97.1	51.9	72.0
WF+NF +POSF	$f1 = w_i$ $f2 = w_{i-1}, w_i, w_{i+1}$ $f3 = pos_i$ $f4 = pos_{i-1}, pos_i, pos_{i+1}$ $f5 = pre\_e_{i-1}$	97.1	51.9	72.0
+PNEF	$f1 = w_i$ $f2 = w_{i-1}, w_i, w_{i+1}$ $f3 = pos_i$ $f4 = pos_{i-1}, pos_i, pos_{i+1}$ $f5 = pre\_e_0, ..., pre\_e_{i-1}$	97.1	52.4	72.2
WF+NF +POSF +PNEF +PDF +PNF +PCF	$f1 = w_i$ $f2 = w_{i-1}, w_i, w_{i+1}$ $f3 = pos_i$ $f4 = pos_{i-1}, pos_i, pos_{i+1}$ $f5 = pre\_e_0, ..., pre\_e_{i-1}$ $f6 = ?(w_{i-1}$ *is a degree word*$)$ $f7 = ?(w_{i-1}$ *is a negative word*$)$ $f8 = ?(w_{i-1}$ *is a conjunction*$)$	97.1	53.0	72.7

4 Encoding Feature: Emotion Components of Word

Researches on the psychology of concepts show that categories in the human mind are not simply sets with clearcut boundaries (Murphy, 2002; Hampton, 2007). Word emotions are certainly related to mental concepts. As for emotion states, most theorists appear to take a combinatorial view. Plutchik (1962), for example, talks about "mixed states", "dyads" and "triads" of primary emotions. Similarly, Averill (1975) argues for compound emotions based on more elementary ones. And one model, suggested by Ekman (1982) (emotion blends) and Plutchik (mixed states), is that emotions mix (Ortony, 1988). According to these researches, we use an encoding feature: emotion components of word.

"Emotion components of word" describes the combined basic emotions in a word, which is represented by eight binary digits, and each digit corresponding to a basic emotion class respectively. For example, the word "喜欢 (like)", its possible emotion components in a certain context is "01100000", which expresses the combined emotions by joy and love.

With the expression of emotion components of word, it is possible to distinguish direct emotion words and indirect emotion words. Those words always demonstrate similar emotion components in different contexts can be regarded as direct emotion words, accordingly, those words demonstrate different emotion components in different contexts can be regarded as indirect emotion words. With the expression of emotion components in word, the problem of expressing emotion ambiguity in words can be solved. The same word in different contexts may reflect different emotions, which can be expressed by different emotion components. The emotions of words with multiple emotions also can be expressed by emotion components.

Table 3: F-value for combined contextual features (CF) and semantic feature (SF) (including sentence class feature (SCF) and word class feature (WCF))

Feature type	Cluster number	F-value (a)	(b)	(c)
CF+SCF	20	97.0	53.1	72.8
	40	97.0	53.4	72.7
	60	97.0	53.5	72.8
	80	97.0	52.9	72.5
	100	97.0	**53.6**	**73.1**
	120	97.0	53.1	72.7
	150	97.0	53.2	72.9
	180	97.0	53.4	73.1
CF+WCF	40	97.0	53.1	72.8
	100	97.0	**53.4**	**72.9**
	160	97.0	**53.4**	**72.9**
	220	97.0	53.3	72.9
	280	97.0	53.2	72.8
	370	97.0	53.1	72.8

The statistics of word emotion components in Ren-CECps show that there are a total of 68 emotion components in all of 22,095 annotated emotion words without repetitions. Figure 1 shows the growth curve of word emotion components number with emotion word number increase.

As can be seen from figure 1, the number increase of word emotion components shows a very slow growth rate with the number increase of emotion words. We can conclude that the space of word emotion components is a relatively small space.

In the model of MaxEnt based Chinese word emotion recognition, the Pre-N-words Emotion Feature (PNEF) and emotion output can be encoded to emotion components.

Pre-N-words Emotion Components Feature (PNECF): The emotion components of its previous words for a word (w_i). The value of this feature is obtained only after the computation of the emotion components for its previous words.

Table 4 gives the results of F-value for the combined contextual features and encoding feature.

As can be seen in table 4, when Pre-N-words Emotion Feature (PNEF) is replaced by Pre-N-words Emotion Components Feature (PNECF), and emotion components are output as results, F-value is increased up to (a=97.3, b=57.3, c=73.3). Then based on this result, we firstly trained a word emotion based model, then the word emotion outputs of this model are used as Pre-N-words Emotion Feature (PNEF) for the word emotion components based model. A significant F-value improvement of task (b) and (c) (b=62.5, c=73.7) over only using contextual and semantic features was observed after adding the combined word emotion and word emotion components as feature.

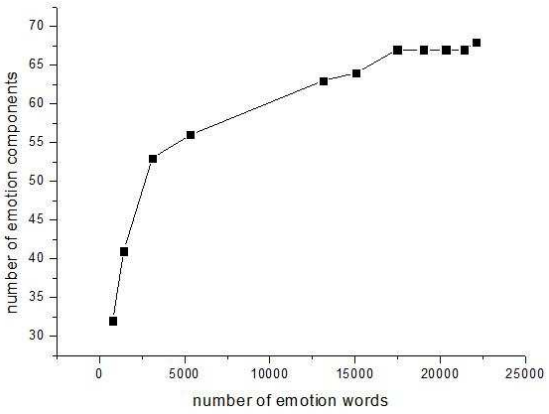


Figure 1: The growth curve of word emotion components

5 Discussion

5.1 Word Emotion Agreement on People's Judgments

The final aim of a human-computer interaction recognition system is to get the result close to people's judgments. As word emotion is inherently uncertain and subjective, here we report the annotation agreement on word emotion of Ren-CECps, which can be taken as an evaluation criteria for a algorithm.

To measure the annotation agreement of Ren-CECps, three annotators independently annotated 26 documents with a total of 805 sentences, 19,738 words. We use the following two metrics to measure agreement on word emotion annotation.

(1) Kappa coefficient of agreement (Carletta, 1996). It is a statistic adopted by the computa-

Table 4: F-value for the combined contextual features and encoding feature

Feature type	Features	F-value		
		(a)	(b)	(c)
WF+NF+POSF+PNECF +PDF+PNF+PCF	$f1 = w_i$ $f2 = w_{i-1}, w_i, w_{i+1}$ $f3 = pos_i$ $f4 = pos_{i-1}, pos_i, pos_{i+1}$ $f5 = pre\_es_0, ..., pre\_es_{i-1}$ $f6 =?(w_{i-1}$ *is a degree word*$)$ $f7 =?(w_{i-1}$ *is a negative word*$)$ $f8 =?(w_{i-1}$ *is a conjunction*$)$	97.3	57.3	73.3
WF+NF+POSF+PNEF +PNECF+PDF+PNF+PCF	$f1 = w_i$ $f2 = w_{i-1}, w_i, w_{i+1}$ $f3 = pos_i$ $f4 = pos_{i-1}, pos_i, pos_{i+1}$ $f5 = pre\_e_0, ..., pre\_e_{i-1}$ $f6 = pre\_es_0, ..., pre\_es_{i-1}$ $f7 =?(w_{i-1}$ *is a degree word*$)$ $f8 =?(w_{i-1}$ *is a negative word*$)$ $f9 =?(w_{i-1}$ *is a conjunction*$)$	97.3	62.5	73.7

tional linguistics community as a standard measure.

(2) Voting agreement. It is used to measure how much intersection there is between the sets of word emotions identified by the annotators. It includes majority-voting agreement ($Agreement_{MV}$) and all-voting agreement ($Agreement_{AV}$). $Agreement_{MV}$ is defined as follows. Let A, B and C be the sets of word emotion components annotated by annotators a, b and c respectively. The expert coder is the set of expressions that agreed by at least two annotators, see equation (4).

$$Agreement_{MV} = Avg(\frac{count(t_i = e_j)}{count(t_i)}) \quad (4)$$

In which, $t_i \in T$, $e_j \in E$, $T = A \bigcup B \bigcup C$, $E = (A \bigcap B) \bigcup (A \bigcap C) \bigcup (B \bigcap C)$.

Accordingly, the expert coder of $Agreement_{AV}$ is the set of expressions that agreed by all annotators.

The above two metrics are used to measure the agreements on: (a) determine if a word is an emotion or unemotion word; (b) determine the eight basic emotions for emotion words (complete emotion matching); (c) determine the eight basic emotions for emotion words (single matching). (b) and (c) are provided that at least two people to believe the word is an emotion word. Table 5 shows the agreements measured by the two metrics.

As shown in table 5, it is easier for annotators to agree at if a word contains emotion, but it is more difficult to agree on emotions or emotion components of a word. Compared with the agreement on people's judgments, our experiments gave promising results.

Table 5: Agreement of word emotion annotation measured by Kappa, Majority-voting (MV), and All-voting (AV)

Measure	Kappa	MV	AV
(a)	84.3	98.5	95.1
(b)	66.7	70.3	26.2
(c)	77.5	100	84.9

5.2 Error Analysis

Conducting an error analysis, we find that a lot of errors occur due to the recognition on multi-emotion words and indirect emotion words, especially in short sentences because the features can be extracted are too few. So more features should be considered from larger contexts, such as the topic emotion of paragraph or document.

There are some errors occur due to more than one emotion holders exist in one sentence, for ex-

ample of sentence (4).

(4) 我发现女儿正看着她感兴趣的玩具。(I found that daughter was looking at the toys of her interest.)

In sentence (4), three annotators all agree that the emotion components of the word "感兴趣 (interest)" is "00000000" since they believe that this word is an unemotion word from the view of the writer. But our system give a result of "00100000" because the emotion holder "女儿 (daughter)" of the emotion word "感兴趣 (interest)" has not been considered in our algorithm. Therefore, the recognition of emotion holder is indispensable for an accurate emotion analysis system.

In addition, Chinese segmentation mistakes and phrasing error also cause errors.

6 Conclusions

Automatically perceive the emotions from text has potentially important applications in CMC (computer-mediated communication) that range from identifying emotions from online blogs to enabling dynamically adaptive interfaces. Therein words play important role in emotion expressions of text.

In this paper we explored features for recognizing word emotions in sentences. Different from previous researches on textual emotion recognition that based on affective lexicons, we believe that besides obvious emotion words referring to emotions, there are words can potentially convey emotions act only as an indirect reference. Also, quite often words that bear emotion ambiguity and multiple emotions are difficult to be recognized depending on emotion lexicons. Emotion of a word should be determined with its context.

Based on Ren-CECps (an annotated emotion corpus) and MaxEnt (Maximum entropy) model, we have experimented several contextual features and their combination, then using PLSA (probabilistic latent semantic analysis), semantic feature are demonstrated the effectiveness for word emotion recognition. A significant performance improvement over only using contextual and semantic features was observed after adding encoding feature (word emotion components). Determining intensity of word emotion and recognizing emotion of sentence or document based on word emotion are included in our future work.

Acknowledgments

This research has been partially supported by Ministry of Education, Science, Sprots and Culture, Grant-in-Aid for Challenging Exploratory Research, 21650030. We also wish to acknowledge the anonymous reviewer's insightful comments and suggestions.

References

J. R. Averill. 1975. A semantic atlas of emotional concepts. *JSAS Catalog of Selected Documents in Psychology*.

Adam Berger, Vincent Della Pietra and Stephen A. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistic* 22(1), pages 39 – 71.

Plaban Kumar Bhowmick, Animesh Mukherjee, Aritra Banik, Pabitra Mitra, Anupam Basu. 2008. A comparative study of the properties of emotional and non-Emotional words in the Wordnet: A complex network approach. In *Proceedings of International conference on natural language processing (ICON 2008)*.

Jean Carletta. 1996. Assessing agreement on classification tasks: the Kappa statistic. *Computational Linguistics*. 22(2): 249-254.

Z. Dong and Q. Dong. 2003. HowNet—a hybrid language and knowledge resource. In *Proceedings of Int'l Conf. Natural Language Processing and Knowledge Eng.*, pages 820 – 824.

Paul Ekman. 1982. *Emotion in the human face*. Cambridge University Press.

Andrea Esuli and Fabrizio Sebastiani. 2006. SentiWordNet: A publicly available lexical resource for opinion mining. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC 2006)*, pages 417-422.

James A. Hampton. 2007. Typicality, graded membership, and vagueness. *Cognitive Science* 31:355 – 384.

Thomas Hofmann. 1999. Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI'99)*.

Abe Kazemzadeh, Sungbok Lee, and Shrikanth Narayanan. 2008. An interval type-2 fuzzy logic system to translate between emotion-related ocabularies. In *Proceedings of Interspeech*.

Lun-Wei Ku, Yu-Ting Liang and Hsin-Hsi Chen. 2006. Tagging heterogeneous evaluation corpora for opinionated tasks. In *Proceedings of Conference on Language Resources and Evaluation (LREC 2006)*, pages 667-670.

Chul Min Lee, Shrikanth S. Narayanan. 2005. Toward detecting emotions in spoken dialogs. *Journal of the American Society for Information Science. IEEE Trans. on Speech and Audio Processing* 13(2):293-303.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of statistical natural language processing*. Cambridge, MA: MIT Press.

Gregory L. Murphy. 2002. *The Big Book of Concepts*. Cambridge, MA: MIT Press.

Andrew Ortony. Gerald l. Clore. Mark A. Foss. 1987. The referential structure of the affective lexicon. *Cognitive Science* 11:341-364.

Andrew Ortony, Gerald L. Clore, Allan Collins. 1988. *The Cognitive Structure of Emotions*. Cambridge University Press.

Robert Plutchik. 1962. *The emotions: Facts, theories, and a new model*. New York: Random House.

Changqin Quan and Fuji Ren. 2010. A blog emotion corpus for emotional expression analysis in Chinese. *Computer Speech & Language*, 24(4):726 – 749.

Philip J. Stone, Dexter C. Dunphy, Marshall S. Smith, and Daniel M. Ogilvie. 1966. *The General Inquirer: A computer approach to content analysis.* The MIT Press.

Carlo Strapparava and Alessandro Valitutti. 2004. Wordnet-affect: an affective extension of wordnet. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC 2004)*, pages 1083-1086.

Carlo Strapparava, Alessandro Valitutti, and Oliviero Stock. 2006. The affective weight of lexicon. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC 2006)*, pages 423-426.

Carlo Strapparava, Alessandro Valitutti, Oliviero Stock. 2007. Dances with words. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 1719 – 1724.

Hiroya Takamura, Takashi Inui, and Manabu Okumura. Extracting emotional polarity of words using spin model. 2005. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pages 133 – 140.

Ryoko Tokuhisa, Kentaro Inui, Yuji Matsumoto. 2008. Emotion classification using massive examples extracted from the web. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*. pages 881 – 888.

Francisco Virginia and Gervás Pablo. 2006. Exploring the compositionality of emotions in text: word emotions, sentence emotions and sutomated Tagging. In *Proceedings of the AAAI-06 Workshop on Computational Aesthetics: Artificial Intelligence Approaches to Beauty and Happiness*, pages 16 – 20.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2009. Recognizing Contextual Polarity: an exploration of features for phrase-level sentiment analysis. *Computational Linguistics* 35(3): 1 – 34.

Inducing Fine-Grained Semantic Classes via Hierarchical and Collective Classification

Altaf Rahman and **Vincent Ng**
Human Language Technology Research Institute
University of Texas at Dallas
{altaf,vince}@hlt.utdallas.edu

Abstract

Research in named entity recognition and mention detection has typically involved a fairly small number of semantic classes, which may not be adequate if semantic class information is intended to support natural language applications. Motivated by this observation, we examine the under-studied problem of semantic subtype induction, where the goal is to automatically determine which of a set of 92 fine-grained semantic classes a noun phrase belongs to. We seek to improve the standard supervised approach to this problem using two techniques: hierarchical classification and collective classification. Experimental results demonstrate the effectiveness of these techniques, whether or not they are applied in isolation or in combination with the standard approach.

1 Introduction

Semantic class determination refers to the task of classifying a noun phrase (NP), be it a name or a nominal, as one of a set of pre-defined semantic classes. A semantic class classifier is a basic text-processing component in many high-level natural language processing (NLP) applications, including information-extraction (IE) systems and question-answering (QA) systems. In recent years, supervised semantic class determination has been tackled primarily in the context of (1) *coreference resolution* (e.g., Ng (2007), Huang et al. (2009)), where semantic classes are induced and subsequently used to disallow coreference between semantically incompatible NPs, and (2) the *mention detection* task in the ACE evaluations (e.g., Florian et al. (2004; 2006)), where the goal is to identify the boundary of a *mention* (i.e., a noun phrase that belongs to one of the pre-defined ACE semantic classes), its mention type (e.g., pronoun, name), and its semantic class. The output of a mention detector is then used by downstream IE components, which typically include a coreference resolution system and a relation extraction system. Owing in part to its potentially large influence on downstream IE components, accurate semantic class determination is crucial.

Over the years, NLP researchers have focused on a relatively small number of semantic classes in both NE recognition and mention detection: seven classes in the MUC-6 and MUC-7 NE recognition task, four classes in the CoNLL 2002 and 2003 NE recognition shared task, and seven classes in the ACE 2005 mention detection task. Given that one of the uses of semantic class information is to support NLP applications, it is questionable whether this purpose can be adequately served by such a small number of semantic classes. For example, given the question "Which city was the first Olympic Games held in?", it would be helpful for a QA system to know which NEs are cities. However, virtually all of the existing NE recognizers and mention detectors can only determine whether an NE is a location or not.

Our goal in this paper is to tackle the under-studied problem of determining fine-grained semantic classes (henceforth *semantic subtypes*). More specifically, we aim to classify an NP as one of the 92 fine-grained, domain-independent semantic classes that are determined to be useful for supporting the development of QA and

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 931–939,
Beijing, August 2010

IE systems in the ACE and AQUAINT programs. These 92 semantic subtypes have been used to manually annotate the NPs in the *BBN Entity Type Corpus* (Weischedel and Brunstein, 2005). Given the availability of this semantic subtype-annotated corpus, we adopt a supervised machine learning approach to semantic subtype determination. Specifically, given (the boundary of) an NP, we train a classification model to determine which of the 92 semantic subtypes it belongs to.

More importantly, we seek to improve the standard approach to semantic subtype induction described above by proposing two techniques. The first technique, collective classification, aims to address a common weakness in the standard supervised learning paradigm, where a classifier classifies each instance independently of the others and is unable to exploit any relational information between a pair (or a subset) of the instances that may be helpful for classification. The second technique, hierarchical classification, exploits the observation that these 92 semantic subtypes can be grouped into a smaller number of coarse-grained semantic types (henceforth semantic supertypes). With this two-level hierarchy, learning can proceed in a sequential fashion: given an NP, we first determine its semantic supertype and then classify it as one of the semantic subtypes that fall under the predicted supertype in the hierarchy. Empirical results show that these two techniques, when applied in isolation to the standard learning approach to subtype induction, can significantly improve its accuracy, and the best result is achieved when they are applied in combination.

The rest of the paper is organized as follows. Section 2 provides an overview of the 92 semantic subtypes and the evaluation corpus. In Section 3, we present our baseline semantic subtype classification system. Sections 4 and 5 introduce collective classification and hierarchical classification respectively, and describe how these two techniques can be used to improve the baseline semantic subtype classifier. We show evaluation results in Section 6 and conclude in Section 7.

2 Semantic Subtypes

As noted before, each name and nominal in the *BBN Entity Type Corpus* is annotated with one of the 92 semantic subtypes. In our experiments, we use all the 200 Penn Treebank Wall Street Journal articles in the corpus, yielding 17,292 NPs that are annotated with their semantic subtypes.

Table 1 presents an overview of these subtypes. Since they have been manually grouped into 29 supertypes, we also show the supertypes in the table. More specifically, the first column shows the supertypes, the second column contains a brief description of a supertype, and the last column lists the subtypes that correspond to the supertype in the first column. In cases where a supertype contains only one subtype (e.g., PERSON), the supertype is not further partitioned into different subtypes; for classification purposes, we simply treat the subtype as identical to its supertype (and hence the two always have the same name). A detailed description of these supertypes and subtypes can be found in Weischedel and Brunstein (2005). Finally, we show the class distribution: the parenthesized number after each subtype is the percentage of the 17,292 NPs annotated with the subtype.

3 Baseline Classification Model

We adopt a supervised machine learning approach to train our baseline classifier for determining the semantic subtype of an NP. This section describes the details of the training process.

Training corpus. As mentioned before, we use the Wall Street Journal articles in the BBN Entity Type Corpus for training the classifier.

Training instance creation. We create one training instance for each annotated NP, NP_i, which is either a name or a nominal, in each training text. The classification of an instance is its annotated semantic subtype value, which is one of the 92 semantic subtypes. Each instance is represented by a set of 33 features[1], as described below.

1. Mention String (3): Three features are derived from the string of NP_i. Specifically, we employ the NP string as a feature. If NP_i contains more than one token, we create one feature for each of its constituent tokens. Finally, to distinguish the different senses of a nominal, we create

[1] As we will see, since we employ an exponential model, an instance may be represented by fewer than 33 features.

Supertype	Brief Description	Subtypes
PERSON	Proper names of people.	Person (9.2).
PERSON DESC	Any head word of a common noun referring to a person or group of people.	Person Desc (16.8).
NORP	This type is named after its subtypes: nationality, religion, political, etc.	Nationality (2.9), Religion (0.1), Political (0.6), Other (0.1).
FACILITY	Names of man-made structures, including infrastructure, buildings, monuments, camps, farms, mines, ports, etc.	Building (0.1), Bridge (0.02), Airport (0.01), Attraction (0.01), Highway Street (0.05), Other (0.1).
FACILITY DESC	Head noun of a noun phrase describing buildings, bridges, airports, etc.	Building (0.5), Bridge (0.05), Airport (0.01), Highway Street (0.2), Attraction (0.02), Other (0.5).
ORGANIZATION	Names of companies, government agencies, educational institutions and other institutions.	Government (3.6), Corporation (8.3), Political (0.5), Educational (0.3), Hotel (0.04), City (0.01), Hospital (0.01), Religious (0.1), Other (0.7).
ORG DESC	Heads of descriptors of companies, educational institutions and other governments, government agencies, etc.	Government (2.1), Corporation (4.3), Political (0.2), Educational (0.1), Religious (0.1), Hotel (0.1), City (0.01), Hospital (0.02), Other (0.7).
GPE	Names of countries, cities, states, provinces, municipalities, boroughs.	Country (4.2), City (3.2), State Province (1.4), Other (0.1).
GPE DESC	Heads of descriptors of countries, cities, states, provinces, municipalities.	Country (0.8), City (0.3), State Province (0.3), Other (0.1).
LOCATION	Names of locations other than GPEs. E.g., mountain ranges, coasts, borders, planets, geo-coordinates, bodies of water.	River (0.03), Lake Sea Ocean (0.05), Region (0.2), Continent (0.1), Other (0.2).
PRODUCT	Name of any product. It does not include the manufacturer).	Food (0.01), Weapon (0.02), Vehicle (0.2), Other (0.2).
PRODUCT DESC	Descriptions of weapons and vehicles only. Cars, buses, machine guns, missiles, bombs, bullets, etc.	Food (0.01), Weapon (0.2), Vehicle (0.97), Other (0.02).
DATE	Classify a reference to a date or period.	Date (7.99), Duration (1.9), Age (0.5), Other (0.4).
TIME	Any time ending with A.M. or P.M.	Time (0.5).
PERCENT	Percent symbol or the actual word percent.	Percent (2.07).
MONEY	Any monetary value.	Money (2.9).
QUANTITY	Used to classify measurements. E.g., 4 miles, 4 grams, 4 degrees, 4 pounds, etc.	1D (0.11), 2D (0.08), 3D (0.1), Energy (0.01), Speed (0.01), Weight (0.1), Other (0.04).
ORDINAL	All ordinal numbers. E.g., First, fourth.	Ordinal (0.6).
CARDINAL	Numerals that provide a count or quantity.	Cardinal (5.1).
EVENT	Named hurricanes, battles, wars, sports events, and other named events.	War (0.03), Hurricane (0.1), Other (0.24).
PLANT	Any plant, flower, tree, etc.	Plant (0.2).
ANIMAL	Any animal class or proper name of an animal, real or fictional.	Animal (0.7).
SUBSTANCE	Any chemicals, elements, drugs, and foods. E.g., boron, penicillin, plutonium.	Food (1.1), Drug (0.46), Chemical (0.23), Other (0.9).
DISEASE	Any disease or medical condition.	Disease (0.6).
LAW	Any document that has been made into a law. E.g., Bill of Rights, Equal Rights.	Law (0.5).
LANGUAGE	Any named language.	Language (0.2).
CONTACT INFO	Address, phone.	Address (0.01), Phone (0.04).
GAME	Any named game.	Game (0.1).
WORK OF ART	Titles of books, songs and other creations.	Book (0.16), Play (0.04), Song (0.03), Painting (0.01), Other (0.4).

Table 1: The 92 semantic subtypes and their corresponding supertypes.

a feature whose value is the concatenation of the head of NP_i and its WordNet sense number.[2]

2. Verb String (3): If NP_i is governed by a verb, the following three features are derived from the governing verb. First, we employ the string of the governing verb as a feature. Second, we create a feature whose value is the semantic role of the

[2]We employ the sense number that is manually annotated for each NP in the WSJ corpus as part of the OntoNotes project (Hovy et al., 2006).

governing verb.[3] Finally, to distinguish the different senses of the governing verb, we create a feature whose value is the concatenation of the verb and its WordNet sense number.

3. Semantic (5): We employ five semantic features. First, if NP_i is an NE, we create a feature whose value is the NE label of NP_i, as determined by the Stanford CRF-based NE recognizer (Finkel et al., 2005). However, if NP_i is a nominal, we create a feature that encodes the WordNet semantic class of which it is a hyponym, using the manually determined sense of NP_i.[4] Moreover, to improve generalization, we employ a feature whose value is the WordNet synset number of the head noun of a nominal. If NP_i has a governing verb, we also create a feature whose value is the WordNet synset number of the verb. Finally, if NP_i is a nominal, we create a feature based on its *WordNet equivalent concept*. Specifically, for each entity type defined in ACE 2005[5], we create a list containing all the word-sense pairs in WordNet (i.e., synsets) whose glosses are compatible with that entity type.[6] Then, given NP_i and its sense, we use these lists to determine if it belongs to any ACE 2005 entity type. If so, we create a feature whose value is the corresponding entity type.

4. Morphological (8). If NP_i is a nominal, we create eight features: prefixes and suffixes of length one, two, three, and four.

5. Capitalization (4): We create four capitalization features to determine whether NP_i `IsAllCap`, `IsInitCap`, `IsCapPeriod`, and `IsAllLower` (see Bikel et al. (1999)).

6. Gazetteers (8): We compute eight gazetteer-based features, each of which checks whether NP_i is in a particular gazetteer. The eight dictionaries contain pronouns (77 entries), common words and words that are not names (399.6k), person names (83.6k), person titles and honorifics (761), vehicle words (226), location names (1.8k), company names (77.6k), and nouns extracted from WordNet that are hyponyms of PERSON (6.3k).

7. Grammatical (2): We create a feature that encodes the part-of-speech (POS) sequence of NP_i obtained via the Stanford POS tagger (Toutanova et al., 2003). In addition, we have a feature that determines whether NP_i is a nominal or not.

We employ maximum entropy (MaxEnt) modeling[7] for training the baseline semantic subtype classifier. MaxEnt is chosen because it provides a *probabilistic* classification for each instance, which we will need to perform collective classification, as described in the next section.

4 Collective Classification

One weakness of the baseline classification model is that it classifies each instance independently. In particular, the model cannot take into account relationships between them that may be helpful for improving classification accuracy. For example, if two NPs are the same string in a given document, then it is more likely than not that they have the same semantic subtype according to the "one sense per discourse" hypothesis (Gale et al., 1992). Incorporating this kind of *relational* information into the feature set employed by the baseline system is not an easy task, since each feature characterizes only a single NP.

To make use of the relational information, one possibility is to design a new learning procedure. Here, we adopt a different approach: we perform collective classification, or joint probabilistic inference, on the output of the baseline model. The idea is to treat the output for each NP, which is a probability distribution over the semantic subtypes, as its *prior* label/class distribution, and convert it into a *posterior* label/class distribution by exploiting the available relational information as an additional piece of evidence. For this purpose, we will make use of *factor graphs*. In this section, we first give a brief overview of factor graphs[8], and show how they can be used to perform joint

[3] We also employ the semantic role that is manually annotated for each NP in the WSJ corpus in OntoNotes.

[4] The semantic classes we considered are person, location, organization, date, time, money, percent, and object.

[5] The ACE 2005 entity types include person, organization, GPE, facility, location, weapon, and vehicle.

[6] Details of how these lists are constructed can be found in Nicolae and Nicolae (2006).

[7] We use the MaxEnt implementation available at http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html

[8] See Bunescu and Mooney (2004) and Loeliger (2004) for a detailed introduction to factor graphs.

inference for semantic subtype determination.

4.1 Factor Graphs

Factor graphs model optimization problems of an objective function g, which is a real-valued function of n random variables $X_1, ..., X_n$. We assume that g can be decomposed into a product of m *factors*. In other words, $g(X_1, ..., X_n) = f_1(s_1(X_1, ..., X_n)) ... f_m(s_m(X_1, ..., X_n))$, where each factor f_k is a real-valued function of some subset of $X_1, ..., X_n$, denoted as $s_k(X_1, ..., X_n)$. Each f_k can be thought of as a feature function that computes the *compatibility* of an assignment of values to the variables in $s_k(X_1, ..., X_n)$ with respect to a user-defined feature. Hence, a larger function value is more desirable, as it corresponds to a more compatible assignment of values to the variables involved.

A factor graph consists of two types of nodes: variable nodes and factor nodes. Each random variable X_i is represented by a variable node, and each factor f_k is represented by a factor node. Each factor node f_k is connected only to the nodes corresponding to s_k. This results in a bipartite graph, where edges exist only between a variable node and a factor node.

Given this graph, there are several methods for finding an optimal assignment of the random variables $X_1, ..., X_n$ such that the objective function g is maximized. Exact inference using the sum-product algorithm (Kschischang et al., 2001) is possible if there are no cycles in the graph; otherwise a belief propagation algorithm, such as loopy belief propagation (Murphy et al., 1999), can be applied. Although there are no cycles in our factor graphs, we choose to use loopy belief propagation as our inferencer, since it performs approximate inference and is therefore computationally more efficient than an exact inferencer.

4.2 Application to Subtype Inference

To apply joint inference to semantic subtype induction, we create one factor graph for each test document, where each variable node is random variable X_i over the set of semantic subtype labels L and represents an NP, NP_i, in the document. To retain the prior probabilities over the semantic subtype labels $l_q \in L$ obtained from the baseline classification model, each variable node is given a factor $f(X_i) = P(X_i = l_q)$. If no additional factors that model the relation between two nodes/instances are introduced, maximizing the objective function for this graph (by maximizing the product of factors) will find an assignment identical to the one obtained by taking the most probable semantic subtype label assigned to each instance by the baseline classifier.

Next, we exploit the relationship between two random variables. Specifically, we want to encourage the inference algorithm to assign the same label to two variables if there exists a relation between the corresponding NPs that can provide strong evidence that they should receive the same label. To do so, we create a *pairwise* factor node that connects two variable nodes if the aforementioned relation between the underlying NPs is satisfied. However, to implement this idea, we need to address two questions.

First, *which relation between two NPs can provide strong evidence that they have the same semantic subtype?* We exploit the coreference relation. Intuitively, the coreference relation is a reasonable choice, as coreferent entities are likely to have the same semantic subtype. Here, we naively posit two NPs as coreferent if at least one of the following conditions is satisfied: (1) they are the same string after determiners are removed; (2) they are aliases (i.e., one is an acronym or abbreviation of the other); and (3) they are both proper names and have at least one word in common (e.g., "Delta" and "Delta Airlines").[9]

Second, *how can we define a pairwise factor, f_{pair}, so that it encourages the inference algorithm to assign the same label to two nodes?* One possibility is to employ the following definition:

$$\begin{aligned} & f_{pair}(X_i, X_j) \\ = \; & P(X_i = l_p, X_j = l_q), \text{where } \; l_p, l_q \in L \\ = \; & \begin{cases} 1 & \text{if } l_p = l_q \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

In essence, f_{pair} prohibits the assignment of different labels to the two nodes it connects. In our

[9] The third condition can potentially introduce many false positives, positing "Bill Clinton" and "Hillary Clinton" as coreferent, for instance. However, this kind of false positives does not pose any problem for us, since the two NPs involved belong to the same semantic subtype (i.e., PERSON).

experiments, however, we "improve" f_{pair} by incorporating semantic supertype information into its definition, as shown below:

$$\begin{aligned} & f_{pair}(X_i, X_j) \\ = \; & P(X_i = l_p, X_j = l_q), \text{where } \; l_p, l_q \in L \\ = \; & \begin{cases} P_{sup}(sup(l_p)|\text{NP}_i) P_{sup}(sup(l_q)|\text{NP}_j) & \text{if } l_p = l_q \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

In this definition, $sup(l_q)$ is the supertype of l_q according to the semantic type hierarchy shown in Section 2, and $P_{sup}(sup(l_q)|\text{NP}_j)$ is the probability that NP_j belongs to $sup(l_q)$ according to the semantic supertype classification model P_{sup} (see Section 5 for details on how this model can be trained). In essence, we estimate the joint probability by (1) assuming that the two events are independent, and then (2) computing each event using supertype information. Intuitively, this definition allows f_{pair} to favor those label assignments that are more compatible with the predictions of P_{sup}.

After graph construction, we apply an inferencer to compute a marginal probability distribution over the labels for each node/instance in the graph by maximizing the objective function g, and output the most probable label for each instance according to its marginal distribution.

5 Hierarchical Classification

The pairwise factor f_{pair} defined above exploits supertype information in a *soft* manner, meaning that the most probable label assigned to an NP by an inferencer is not necessarily consistent with its predicted supertype (e.g., an NP may receive Hotel as its subtype even if its supertype is PERSON). In this section, we discuss how to use supertype information for semantic subtype classification in a *hard* manner so that the predicted subtype is consistent with its supertype.

To exploit supertype information, we first train a model, P_{sup}, for determining the semantic supertype of an NP using MaxEnt. This model is trained in essentially the same way as the baseline model described in Section 3. In particular, it is trained on the same set of instances using the same feature set as the baseline model. The only difference is that the class value of each training instance is the semantic supertype of the associated NP rather than its semantic subtype.

Next, we train 29 supertype-specific classification models for determining the semantic subtype of an NP. For instance, the ORGANIZATION-specific classification model will be used to classify an NP as belonging to one of its subtypes (e.g., Government, Corporation, Political agencies). A supertype-specific classification model is trained much like the baseline model. Each instance is represented using the same set of features as in the baseline, and its class label is its semantic subtype. The only difference is that the model is only trained only on the subset of the instances for which it is intended. For instance, the ORGANIZATION-specific classification model is trained only on instances whose class is a subtype of ORGANIZATION.

After training, we can apply the supertype classification model and the supertype-specific subtype classification model to determine the semantic subtype of an NP in a hierarchical fashion. Specifically, we first employ the supertype model to determine its semantic supertype. Then, depending on this predicted semantic supertype, we use the corresponding subtype classification model to determine its subtype.

6 Evaluation

For evaluation, we partition the 200 Wall Street Journal Articles in the BBN Entity Type corpus into a training set and a test set following a 80/20 ratio. As mentioned before, each text in the Entity Type corpus has its NPs annotated with their semantic subtypes. Test instances are created from these texts in the same way as the training instances described in Section 3. To investigate whether we can benefit from hierarchical and collective classifications, we apply these two techniques to the Baseline classification model in isolation and in combination, resulting in the four sets of results in Tables 2 and 3.

The Baseline results are shown in the second column of Table 2. Due to space limitations, it is not possible to show the result for each semantic subtype. Rather, we present semantic supertype results, which are obtained by micro-averaging the corresponding semantic subtype results and are expressed in terms of recall (R), precision (P), and F-measure (F). Note that only those semantic

	Semantic Supertype	Baseline only R	Baseline only P	Baseline only F	Baseline+Hierarchical R	Baseline+Hierarchical P	Baseline+Hierarchical F
1	PERSON	91.9	89.7	90.8	88.8	91.1	89.9
2	PERSON DESC	91.3	87.8	89.5	92.1	89.8	91.0
3	SUBSTANCE	60.0	66.7	63.2	70.0	58.3	63.6
4	NORP	87.8	90.3	89.0	91.9	90.7	91.3
5	FACILITY DESC	72.7	88.9	80.0	68.2	93.8	79.0
6	ORGANIZATION	76.6	73.8	75.2	78.5	73.2	75.8
7	ORG DESC	75.0	70.7	72.8	75.8	75.2	75.5
8	GPE	75.6	73.9	74.7	77.0	75.4	76.2
9	GPE DESC	60.0	75.0	66.7	70.0	70.0	70.0
10	PRODUCT DESC	53.3	88.9	66.7	53.3	88.9	66.7
11	DATE	85.0	85.0	85.0	84.5	85.4	85.0
12	PERCENT	100.0	100.0	100.0	100.0	100.0	100.0
13	MONEY	83.9	86.7	85.3	88.7	96.5	92.4
14	QUANTITY	22.2	100.0	36.4	66.7	66.7	66.7
15	ORDINAL	100.0	100.0	100.0	100.0	100.0	100.0
16	CARDINAL	96.0	77.4	85.7	94.0	81.0	87.0
	Accuracy		81.56			82.60	

Table 2: Results for Baseline only and Baseline with hierarchical classification.

	Semantic Supertype	Baseline+Collective R	Baseline+Collective P	Baseline+Collective F	Baseline+Both R	Baseline+Both P	Baseline+Both F
1	PERSON	93.8	98.1	95.9	91.9	100.0	95.8
2	PERSON DESC	93.9	88.5	91.1	92.6	89.5	91.0
3	SUBSTANCE	60.0	85.7	70.6	70.0	63.6	66.7
4	NORP	89.2	93.0	91.0	90.5	94.4	92.4
5	FACILITY DESC	63.6	87.5	73.7	68.2	93.8	79.0
6	ORGANIZATION	85.8	76.2	80.7	87.4	76.3	81.3
7	ORG DESC	75.8	74.1	74.9	75.8	74.6	75.2
8	GPE	74.1	75.8	74.9	81.5	81.5	81.5
9	GPE DESC	60.0	60.0	60.0	70.0	77.8	73.7
10	PRODUCT DESC	53.3	88.9	66.7	53.3	88.9	66.7
11	DATE	85.0	85.4	85.2	85.0	86.3	85.6
12	PERCENT	100.0	100.0	100.0	100.0	100.0	100.0
13	MONEY	83.9	86.7	85.3	90.3	96.6	93.3
14	QUANTITY	22.2	100.0	36.4	66.7	66.7	66.7
15	ORDINAL	100.0	100.0	100.0	100.0	100.0	100.0
16	CARDINAL	96.0	78.7	86.5	94.0	83.9	88.7
	Accuracy		83.70			85.08	

Table 3: Results for Baseline with collective classification and Baseline with both techniques.

supertypes with non-zero scores are shown. As we can see, only 16 of the 29 supertypes have non-zero scores.[10] Among the "traditional" semantic types, the Baseline yields good performance for PERSON, but only mediocre performance for ORGANIZATION and GPE. While additional experiments are needed to determine the reason, we speculate that this can be attributed to the fact that PERSON and PERSON DESC have only one semantic subtype (which is the supertype itself), whereas ORGANIZATION and GPE have nine and four subtypes, respectively. The classification accuracy is shown in the last row of the table. As we can see, the Baseline achieves an accuracy of 81.56.

Results obtained when hierarchical classification is applied to the Baseline are shown in the third column of Table 2. In comparison to the Baseline, accuracy rises from 81.56 to 82.60. This represents an error reduction of 5.6%, and the difference between these two accuracies is statistically significant at the $p = 0.04$ level.[11]

[10] The 13 supertypes that have zero scores are all under-represented classes, each of which accounts for less than one percent of the instances in the dataset.

[11] All significance test results in this paper are obtained using Approximate Randomization (Noreen, 1989).

Results obtained when collective classification alone is applied to the Baseline are shown in the second column of Table 3. In this case, the prior probability distribution over the semantic subtypes that is needed to create the factor associated with each node in the factor graph is simply the probabilistic classification of the test instance that the node corresponds to. In comparison to the Baseline, accuracy rises from 81.56 to 83.70. This represents an error reduction of 11.6%, and the difference is significant at the $p = 0.01$ level. Also, applying collective classification to the Baseline yields slightly better results than applying hierarchical classification to the Baseline, and the difference in their results is significant at the $p = 0.002$ level.

Finally, results obtained when both hierarchical and collective classification are applied to the Baseline are shown in the third column of Table 3. In this case, the prior distribution needed to create the factor associated with each node in the factor graph is provided by the supertype-specific classification model that is used to classify the test instance in hierarchical classification. In comparison to the Baseline, accuracy rises from 81.56 to 85.08. This represents an error reduction of 19.1%, and the difference is highly significant ($p < 0.001$). Also, applying both techniques to the Baseline yields slightly better results than applying only collective classification to the Baseline, and the difference in their results is significant at the $p = 0.003$ level.

6.1 Feature Analysis

Next, we analyze the effects of the seven feature types described in Section 3 on classification accuracy. To measure feature performance, we take the best-performing system (i.e., Baseline combined with both techniques), begin with all seven feature types, and iteratively remove them one by one so that we get the best accuracy. The results are shown in Table 4. Across the top line, we list the numbers representing the seven feature classes. The feature class that corresponds to each number can be found in Section 3, where they are introduced. For instance, "2" refers to the features computed based on the governing verb. The first row of results shows the system performance after removing just one feature class. In this case, removing the sixth feature class (Gazetteers) improves accuracy to 85.6, while removing the mention string features reduces accuracy to 81.4. The second row repeats this, after removing the gazetteer features.

1	3	7	4	2	5	6
81.4	75.8	83.3	83.7	84.1	85.2	85.6
80.4	74.9	84.3	85.3	85.3	86.1	
80.4	78.3	83.9	86.5	86.7		
81.8	76.2	85.2	87.6			
75.4	83.4	84.6				
66.2	80.9					

Table 4: Results of feature analysis.

Somewhat surprisingly, using only mention string, semantic, and grammatical features yields the best accuracy (87.6). This indicates that gazetteers, morphological features, capitalization, and features computed based on the governing verb are not useful. Removing the grammatical features yields a 3% drop in accuracy. After that, accuracy drops by 4% when semantic features are removed, whereas a 18% drop in accuracy is observed when the mention string features are removed. Hence, our analysis suggests that the mention string features are the most useful features for semantic subtype prediction.

7 Conclusions

We examined the under-studied problem of semantic subtype induction, which involves classifying an NP as one of 92 semantic classes, and showed that two techniques — hierarchical classification and collective classification — can significantly improve a baseline classification model trained using an off-the-shelf learning algorithm on the BBN Entity Type Corpus. In particular, collective classification addresses a major weakness of the standard feature-based learning paradigm, where a classification model classifies each instance independently, failing to capture the relationships among subsets of instances that might improve classification accuracy. However, collective classification has not been extensively applied in the NLP community, and we hope that our work can increase the awareness of this powerful technique among NLP researchers.

Acknowledgments

We thank the three anonymous reviewers for their invaluable comments on an earlier draft of the paper. This work was supported in part by NSF Grant IIS-0812261.

References

Bikel, Daniel M., Richard Schwartz, and Ralph M. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning: Special Issue on Natural Language Learning*, 34(1–3):211–231.

Bunescu, Razvan and Raymond J. Mooney. 2004. Collective information extraction with relational markov networks. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 483–445.

Finkel, Jenny Rose, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 363–370.

Florian, Radu, Hany Hassan, Abraham Ittycheriah, Hongyan Jing, Nanda Kambhatla, Xiaoqiang Luo, Nicolas Nicolov, and Salim Roukos. 2004. A statistical model for multilingual entity detection and tracking. In *HLT-NAACL 2004: Main Proceedings*, pages 1–8.

Florian, Radu, Hongyan Jing, Nanda Kambhatla, and Imed Zitouni. 2006. Factorizing complex models: A case study in mention detection. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 473–480.

Gale, William, Ken Church, and David Yarowsky. 1992. One sense per discourse. In *Proceedings of the 4th DARPA Speech and Natural Language Workshop*, pages 233–237.

Hovy, Eduard, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 57–60.

Huang, Zhiheng, Guangping Zeng, Weiqun Xu, and Asli Celikyilmaz. 2009. Accurate semantic class classifier for coreference resolution. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1232–1240.

Kschischang, Frank, Brendan J. Frey, and Hans-Andrea Loeliger. 2001. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47:498–519.

Loeliger, Hans-Andrea. 2004. An introduction to factor graphs. *IEEE Signal Processing Magazine*, 21(1):28–41.

Murphy, Kevin P., Yair Weiss, and Michael I. Jordan. 1999. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 467–475.

Ng, Vincent. 2007. Semantic class induction and coreference resolution. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 536–543.

Nicolae, Cristina and Gabriel Nicolae. 2006. BestCut: A graph algorithm for coreference resolution. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 275–283.

Noreen, Eric W. 1989. *Computer Intensive Methods for Testing Hypothesis: An Introduction*. John Wiley & Sons.

Toutanova, Kristina, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *HLT-NAACL 2003: Proceedings of the Main Conference*, pages 173–180.

Weischedel, Ralph and Ada Brunstein. 2005. BBN pronoun coreference and entity type corpus. In Linguistic Data Consortium, Philadelphia.

Fast, Greedy Model Minimization for Unsupervised Tagging

Sujith Ravi and **Ashish Vaswani** and **Kevin Knight** and **David Chiang**
University of Southern California
Information Sciences Institute
{sravi,avaswani,knight,chiang}@isi.edu

Abstract

Model minimization has been shown to work well for the task of unsupervised part-of-speech tagging with a dictionary. In (Ravi and Knight, 2009), the authors invoke an integer programming (IP) solver to do model minimization. However, solving this problem exactly using an integer programming formulation is intractable for practical purposes. We propose a novel two-stage greedy approximation scheme to replace the IP. Our method runs fast, while yielding highly accurate tagging results. We also compare our method against standard EM training, and show that we consistently obtain better tagging accuracies on test data of varying sizes for English and Italian.

1 Introduction

The task of unsupervised part-of-speech (POS) tagging with a dictionary as formulated by Merialdo (1994) is: given a raw word sequence and a dictionary of legal POS tags for each word type, tag each word token in the text. A common approach to modeling such sequence labeling problems is to build a bigram Hidden Markov Model (HMM) parameterized by *tag-bigram* transition probabilities $P(t_i|t_{i-1})$ and *word-tag* emission probabilities $P(w_i|t_i)$. Given a word sequence w and a tag sequence t, of length N, the joint probability $P(w,t)$ is given by:

$$P(w,t) = \prod_{i=1}^{N} P(w_i|t_i) \cdot P(t_i|t_{i-1}) \quad (1)$$

We can train this model using the Expectation Maximization (EM) algorithm (Dempster and Rubin, 1977) which learns $P(w_i|t_i)$ and $P(t_i|t_{i-1})$ that maximize the likelihood of the observed data. Once the parameters are learnt, we can find the best tagging using the Viterbi algorithm.

$$\hat{t} = \arg\max_{t} P(w,t) \quad (2)$$

Ravi and Knight (2009) attack the Merialdo task in two stages. In the first stage, they search for a minimized transition model (i.e., the smallest set of tag bigrams) that can explain the data using an integer programming (IP) formulation. In the second stage, they build a smaller HMM by restricting the transition parameters to only those tag bigrams selected in the minimization step. They employ the EM algorithm to train this model, which prunes away some of the emission parameters. Next, they use the pruned emission model along with the original transition model (which uses the full set of tag bigrams) and retrain using EM. This alternating EM training procedure is repeated until the number of tag bigrams in the Viterbi tagging output does not change between subsequent iterations. The final Viterbi tagging output from their method achieves state-of-the-art accuracy for this task. However, their minimization step involves solving an integer program, which can be very slow, especially when scaling to large-scale data and more complex tagging problems which use bigger tagsets. In this paper, we present a novel method that optimizes the same objective function using a fast greedy model selection strategy. Our contributions are summarized below:

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 940–948,
Beijing, August 2010

- We present an efficient two-phase greedy-selection method for solving the minimization objective from Ravi and Knight (2009), which runs much faster than their IP.

- Our method easily scales to large data sizes (and big tagsets), unlike the previous minimization-based approaches and we show runtime comparisons for different data sizes.

- We achieve very high tagging accuracies comparable to state-of-the-art results for unsupervised POS tagging for English.

- Unlike previous approaches, we also show results obtained when testing on the entire Penn Treebank data (973k word tokens) in addition to the standard 24k test data used for this task. We also show the effectiveness of this approach for Italian POS tagging.

2 Previous work

There has been much work on the unsupervised part-of-speech tagging problem. Goldwater and Griffiths (2007) also learn small models employing a fully Bayesian approach with sparse priors. They report 86.8% tagging accuracy with manual hyperparameter selection. Smith and Eisner (2005) design a *contrastive estimation* technique which yields a higher accuracy of 88.6%. Goldberg et al. (2008) use linguistic knowledge to initialize the the parameters of the HMM model prior to EM training. They achieve 91.4% accuracy. Ravi and Knight (2009) use a Minimum Description Length (MDL) method and achieve the best results on this task thus far (91.6% word token accuracy, 91.8% with random restarts for EM). Our work follows a similar approach using a model minimization component and alternate EM training.

Recently, the integer programming framework has been widely adopted by researchers to solve other NLP tasks besides POS tagging such as semantic role labeling (Punyakanok et al., 2004), sentence compression (Clarke and Lapata, 2008), decipherment (Ravi and Knight, 2008) and dependency parsing (Martins et al., 2009).

3 Model minimization formulated as a Path Problem

The complexity of the model minimization step in (Ravi and Knight, 2009) and its proposed approximate solution can be best understood if we formulate it as a path problem in a graph.

Let $w = w_0, w_1, \ldots, w_N, w_{N+1}$ be a word sequence where $w_1, \ldots, w_N$ are the input word tokens and $\{w_0, w_{N+1}\}$ are the $start/end$ tokens. Let $T = \{T_1, \ldots, T_K\} \bigcup \{T_0, T_{K+1}\}$ be the fixed set of all possible tags. T_0 and T_{K+1} are special tags that we add for convenience. These would be the $start$ and end tags that one typically adds to the HMM lattice. The tag dictionary D contains entries of the form (w_i, T_j) for all the possible tags T_j that word token w_i can have. We add entries (w_0, T_0) and (w_{K+1}, T_{K+1}) to D. Given this input, we now create a directed graph $G(V, E)$. Let $C_0, C_1 \ldots, C_{K+1}$ be columns of nodes in G, where column C_i corresponds to word token w_i. For all $i = 0, \ldots, N+1$ and $j = 0, \ldots, K+1$, we add node $C_{i,j}$ in column C_i if $(w_i, T_j) \in D$. Now, $\forall i = 0, \ldots, N$, we create directed edges from every node in C_i to every node in C_{i+1}. Each of these edges $e = (C_{i,j}, C_{i+1,k})$ is given the label (T_j, T_k) which corresponds to a tag bigram. This creates our directed graph. Let $l(e)$ be the tag bigram label of edges $e \in E$. For every path P from $C_{0,0}$ to $C_{N+1,K+1}$, we say that P uses an edge label or tag bigram (T_j, T_k) if there exists an edge e in P such that $l(e) = (T_j, T_k)$. We can now formulate the the optimization problem as: *Find the smallest set S of tag bigrams such that there exists at least one path from $C_{0,0}$ to $C_{N+1,K+1}$ using only the tag bigrams in S.* Let us call this the Minimal Tag Bigram Path (MinTagPath) problem.

Figure 1 shows an example graph where the input word sequence is $w_1, \ldots, w_4$ and $T = \{T_1, \ldots, T_3\}$ is the input tagset. We add the start/end word tokens $\{w_0, w_5\}$ and corresponding tags $\{T_0, T_4\}$. The edges in the graph are instantiated according to the word/tag dictionary D provided as input. The node and edge labels are also illustrated in the graph. Our goal is to find a path from $C_{0,0}$ to $C_{5,4}$ using the smallest set of tag bigrams.

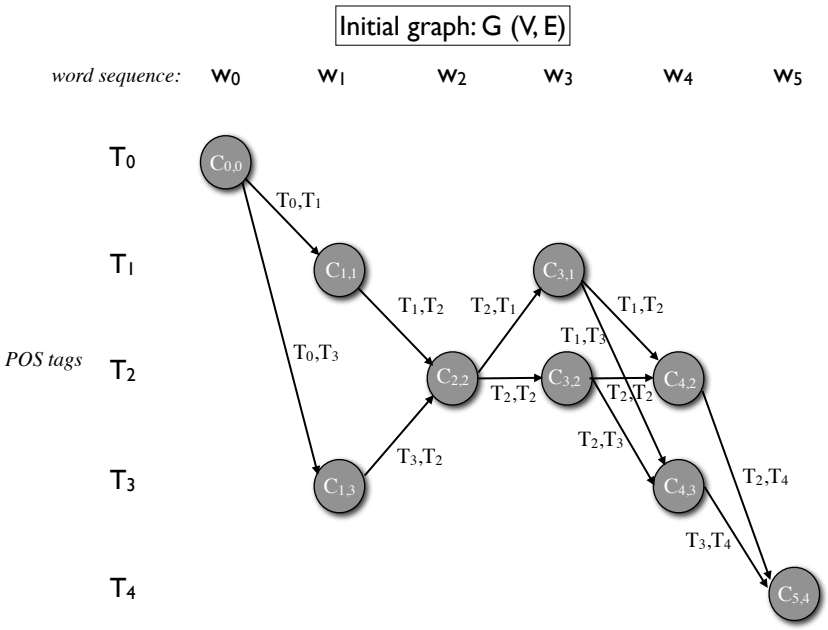


Figure 1: Graph instantiation for the MinTagPath problem.

4 Problem complexity

Having defined the problem, we now show that it can be solved in polynomial time even though the number of paths from $C_{0,0}$ to $C_{N+1,K+1}$ is exponential in N, the input size. This relies on the assumption that the tagset T is fixed in advance, which is the case for most tagging tasks.[1] Let B be the set of all the tag bigram labels in the graph, $B = \{l(e), \forall e \in E\}$. Now, the size of B would be at most $K^2 + 2K$ where every word could be tagged with every possible tag. For $m = 1 \ldots |B|$, let B^m be the set of subsets of B each of which have size m. Algorithm 1 optimally solves the MinTagPath problem.

Algorithm 1 basically enumerates all the possible subsets of B, from the smallest to the largest, and checks if there is a path. It exits the first time a path is found and therefore finds the smallest possible set s_i of size m such that a path exists that uses only the tag bigrams in s_i. This implies the correctness of the algorithm. To check for path existence, we could either throw away all the edges from E not having a label in s_i, and then execute a Breadth-First-Search (BFS) or we could traverse only the edges with labels in s_i during BFS. The running time of Algorithm 1 is easy to calculate. Since, in the worst case we go over all the subsets of size $m = 1, \ldots, |B|$ of B, the number of iterations we can perform is at most $2^{|B|}$, the size of the powerset $\mathcal{P}$ of B. In each iteration, we do a BFS through the lattice, which has $O(N)$ time complexity[2] since the lattice size is linear in N and BFS is linear in the lattice size. Hence the running time is $\leq 2^{|B|} \cdot O(N) = O(N)$. Even though this shows that MinTagPath can be solved in polynomial time, the time complexity is prohibitively large. For the Penn Treebank, $K = 45$ and the

Algorithm 1 Brute Force solution to MinTagPath

```
for m = 1 to |B| do
  for s_i ∈ B^m do
    Use Breadth First Search (BFS) to check
    if ∃ path P from C_{0,0} to C_{N+1,K+1} using
    only the tag bigrams in s_i.
    if P exists then
      return s_i, m
    end if
  end for
end for
```

[1] If K, the size of the tagset, is a variable as well, then we suspect the problem is NP-hard.

[2] Including throwing away edges or not.

worst case running time would be $\approx 10^{13.55} \cdot N$. Clearly, for all practical purposes, this approach is intractable.

5 Greedy Model Minimization

We do not know of an efficient, exact algorithm to solve the MinTagPath problem. Therefore, we present a simple and fast two-stage greedy approximation scheme. Notice that an optimal path P (or any path) *covers* all the input words i.e., every word token w_i has one of its possible taggings in P. Exploiting this property, in the first phase, we set our goal to cover all the word tokens using the least possible number of tag bigrams. This can be cast as a set cover problem (Garey and Johnson, 1979) and we use the set cover greedy approximation algorithm in this stage. The output tag bigrams from this phase might still not allow any path from $C_{0,0}$ to $C_{N+1,K+1}$. So we carry out a second phase, where we greedily add a few tag bigrams until a path is created.

5.1 Phase 1: Greedy Set Cover

In this phase, our goal is to cover all the word tokens using the least number of tag bigrams. The covering problem is exactly that of set cover. Let $U = \{w_0, \ldots, w_N + 1\}$ be the set of elements that needs to be covered (in this case, the word tokens). For each tag bigram $(T_i, T_j) \in B$, we define its corresponding covering set S_{T_i,T_j} as follows:

$$\begin{aligned} S_{T_i,T_j} \;=\; \{w_n \;:\; & ((w_n, T_i) \in D \\ & \wedge (C_{n,i}, C_{n+1,j}) \in E \\ & \wedge l(C_{n,i}, C_{n+1,j}) = (T_i, T_j)) \\ & \bigvee \; ((w_n, T_j) \in D \\ & \wedge (C_{n-1,i}, C_{n,j}) \in E \\ & \wedge l(C_{n-1,i}, C_{n,j}) = (T_i, T_j))\} \end{aligned}$$

Let the set of covering sets be X. We assign a cost of 1 to each covering set in X. The goal is to select a set $CHOSEN \subseteq X$ such that $\bigcup_{S_{T_i,T_j} \in CHOSEN} = U$, minimizing the total cost of $CHOSEN$. This corresponds to covering all the words with the least possible number of tag bigrams. We now use the greedy approximation algorithm for set cover to solve this problem. The pseudo code is shown in Algorithm 2.

Algorithm 2 Set Cover : Phase 1

Definitions
Define $CAND$: Set of candidate covering sets in the current iteration
Define U_{rem} : Number of elements in U remaining to be covered
Define $E_{S_{T_i,T_j}}$: Current effective cost of a set
Define Itr : Iteration number

Initializations
LET $CAND = X$
LET $CHOSEN = \emptyset$
LET $U_{rem} = U$
LET $Itr = 0$
LET $E_{S_{T_i,T_j}} = \frac{1}{|S_{T_i,T_j}|}, \forall\, S_{T_i,T_j} \in CAND$

while $U_{rem} \neq \emptyset$ **do**
$Itr \leftarrow Itr + 1$
Define $\hat{S}_{Itr} = \underset{S_{T_i,T_j} \in CAND}{\operatorname{argmin}} E_{S_{T_i,T_j}}$
$CHOSEN = CHOSEN \bigcup \hat{S}_{Itr}$
Remove $\hat{S}_{Itr}$ from $CAND$
Remove all the current elements in $\hat{S}_{Itr}$ from U_{rem}
Remove all the current elements in $\hat{S}_{Itr}$ from every $S_{T_i,T_j} \in CAND$
Update effective costs, $\forall\, S_{T_i,T_j} \in CAND$, $E_{S_{T_i,T_j}} = \frac{1}{|S_{T_i,T_j}|}$
end while
return $CHOSEN$

For the graph shown in Figure 1, here are a few possible covering sets S_{T_i,T_j} and their initial effective costs $E_{S_{T_i,T_j}}$.

- $S_{T_0,T_1} = \{w_0, w_1\}$, $E_{S_{T_0,T_1}} = 1/2$
- $S_{T_1,T_2} = \{w_1, w_2, w_3, w_4\}$, $E_{S_{T_1,T_2}} = 1/4$
- $S_{T_2,T_2} = \{w_2, w_3, w_4\}$, $E_{S_{T_2,T_2}} = 1/3$

In every iteration Itr of Algorithm 2, we pick a set $\hat{S}_{Itr}$ that is most cost effective. The elements that $\hat{S}_{Itr}$ covers are then removed from all the remaining candidate sets and U_{rem} and the effectiveness of the candidate sets is recalculated for the next iteration. The algorithm stops when all elements of U i.e., all the word tokens are covered. Let, $B_{CHOSEN} = \{(T_i, T_j) : S_{T_i,T_j} \in$

$CHOSEN\}$, be the set of tag bigrams that have been chosen by set cover. Now, we check, using BFS, if there exists a path from $C_{0,0}$ to $C_{N+1,K+1}$ using only the tag bigrams in B_{CHOSEN}. If not, then we have to add tag bigrams to B_{CHOSEN} to enable a path. To accomplish this, we carry out the second phase of this scheme with another greedy strategy (described in the next section).

For the example graph in Figure 1, one possible solution $B_{CHOSEN} = \{(T_0, T_1), (T_1, T_2), (T_2, T_4)\}$.

5.2 Phase 2: Greedy Path Completion

We define a graph $G_{CHOSEN}(V', E') \subseteq G(V, E)$ that contains the edges $e \in E$ such $l(e) \in B_{CHOSEN}$.

Let $B_{CAND} = B \setminus B_{CHOSEN}$, be the current set of candidate tag bigrams that can be added to the final solution which would create a path. We would like to know how many $holes$ a particular tag bigram (T_i, T_j) can fill. We define a hole as an edge e such that $e \in G \setminus G_{CHOSEN}$ and there exists $e', e'' \in G_{CHOSEN}$ such that $tail(e') = head(e) \wedge tail(e) = head(e'')$.

Figure 2 illustrates the graph G_{CHOSEN} using tag bigrams from the example solution to Phase 1 (Section 5.1). The dotted edge $(C_{2,2}, C_{3,1})$ represents a hole, which has to be filled in the current phase in order to complete a path from $C_{0,0}$ to $C_{5,4}$.

In Algorithm 3, we define the effectiveness of a candidate tag bigram $H(T_i, T_j)$ to be the number of holes it covers. In every iteration, we pick the most effective tag bigram, fill the holes and recalculate the effectiveness of the remaining candidate tag bigrams.

Algorithm 3 returns B_{FINAL}, the final set of chosen tag bigrams. It terminates when a path has been found.

5.3 Fitting the Model

Once the greedy algorithm terminates and returns a minimized grammar of tag bigrams, we follow the approach of Ravi and Knight (2009) and fit the minimized model to the data using the alternating EM strategy. The alternating EM iterations are terminated when the change in the size of the observed grammar (i.e., the number of unique tag bigrams in the tagging output) is $\leq 5\%$. We refer to our entire approach using greedy minimization followed by EM training as MIN-GREEDY.

Algorithm 3 Greedy Path Complete : Phase 2

Define B_{FINAL} : Final set of tag bigrams selected by the two-phase greedy approach

LET $B_{FINAL} = B_{CHOSEN}$
LET $H(T_i, T_j) = |\{e\}|$ such that $l(e) = (T_i, T_j)$ and e is a hole, $\forall\ (T_i, T_j) \in B_{CAND}$

while $\nexists$ path P from $C_{0,0}$ to $C_{N+1,K+1}$ using only $(T_i, T_j) \in B_{CHOSEN}$ **do**
  Define $(\hat{T}_i, \hat{T}_j) = \underset{(T_i,T_j) \in B_{CAND}}{\operatorname{argmax}} H(T_i, T_j)$
  $B_{FINAL} = B_{FINAL} \bigcup (\hat{T}_i, \hat{T}_j)$
  Remove $(\hat{T}_i, \hat{T}_j)$ from B_{CAND}
  $G_{CHOSEN} = G_{CHOSEN} \bigcup \{e\}$ such that $l(e) = (T_i, T_j)$
  $\forall\ (T_i, T_j) \in B_{CAND}$, Recalculate $H(T_i, T_j)$
end while
return B_{FINAL}

6 Experiments and Results

6.1 English POS Tagging

Data: We use a standard test set (consisting of 24,115 word tokens from the Penn Treebank) for the POS tagging task (described in Section 1). The tagset consists of 45 distinct tag labels and the dictionary contains 57,388 word/tag pairs derived from the entire Penn Treebank. Per-token ambiguity for the test data is about 1.5 tags/token. In addition to the standard 24k dataset, we also train and test on larger data sets of 48k, 96k, 193k, and the entire Penn Treebank (973k).

Methods: We perform comparative evaluations for POS tagging using three different methods:

1. **EM**: Training a bigram HMM model using EM algorithm.
2. **IP**: Minimizing grammar size using integer programming, followed by EM training (Ravi and Knight, 2009).
3. **MIN-GREEDY**: Minimizing grammar size using the Greedy method described in Sec-

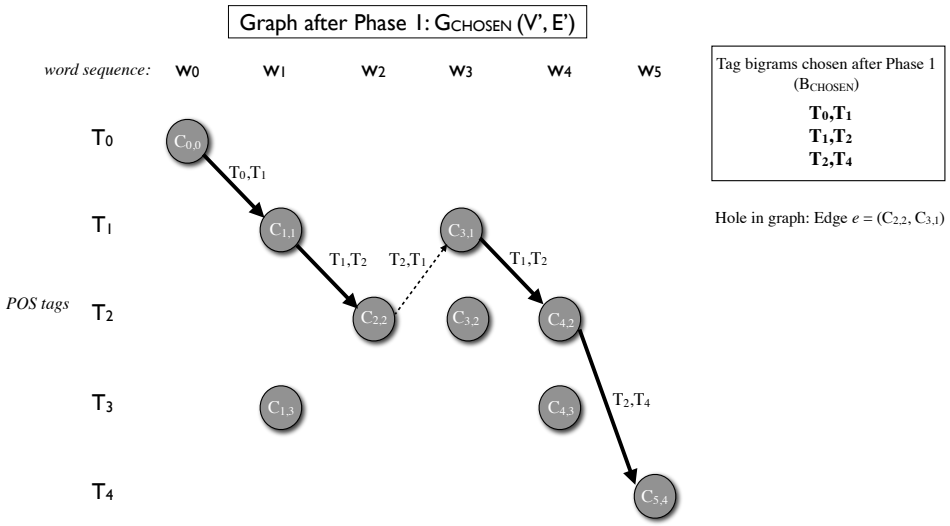


Figure 2: Graph constructed with tag bigrams chosen in Phase 1 of the MIN-GREEDY method.

tion 5, followed by EM training.

Results: Figure 3 shows the tagging performance (word token accuracy %) achieved by the three methods on the standard test (24k tokens) as well as Penn Treebank test (PTB = 973k tokens). On the 24k test data, the MIN-GREEDY method achieves a high tagging accuracy comparable to the previous best from the IP method. However, the IP method does not scale well which makes it infeasible to run this method in a much larger data setting (the entire Penn Treebank). MIN-GREEDY on the other hand, faces no such problem and in fact it achieves high tagging accuracies on all four datasets, consistently beating EM by significant margins. When tagging all the 973k word tokens in the Penn Treebank data, it produces an accuracy of 87.1% which is much better than EM (82.3%) run on the same data.

Ravi and Knight (2009) mention that it is possible to interrupt the IP solver and obtain a suboptimal solution faster. However, the IP solver did not return any solution when provided the same amount of time as taken by MIN-GREEDY for any of the data settings. Also, our algorithms were implemented in Python while the IP method employs the best available commercial software package (CPLEX) for solving integer programs.

Figure 4 compares the running time efficiency for the IP method versus MIN-GREEDY method as we scale to larger datasets. Since the IP solver shows variations in running times for different datasets of the same size, we show the average running times for both methods (for each row in the figure, we run a particular method on three different datasets with similar sizes and average the running times). The figure shows that the greedy approach can scale comfortably to large data sizes, and a complete run on the entire Penn Treebank data finishes in just 1485 seconds. In contrast, the IP method does not scale well—on average, it takes 93 seconds to finish on the 24k test (versus 34 seconds for MIN-GREEDY) and on the larger PTB test data, the IP solver runs for

Test set	**Efficiency** (average running time in secs.)	
	IP	MIN-GREEDY
24k test	93.0	34.3
48k test	111.7	64.3
96k test	397.8	93.3
193k test	2347.0	331.0
PTB (973k) test	*	1485.0

Figure 4: Comparison of MIN-GREEDY versus MIN-GREEDY approach in terms of *efficiency* (average running time in seconds) for different data sizes. All the experiments were run on a single machine with a 64-bit, 2.4 GHz AMD Opteron 850 processor.

Method	Tagging accuracy (%) when training & testing on:				
	24k	*48k*	*96k*	*193k*	*PTB (973k)*
EM	81.7	81.4	82.8	82.0	82.3
IP	91.6	89.3	89.5	91.6	*
MIN-GREEDY	91.6	88.9	89.4	89.1	87.1

Figure 3: Comparison of tagging accuracies on test data of varying sizes for the task of unsupervised English POS tagging with a dictionary using a 45-tagset. *(* IP method does not scale to large data).*

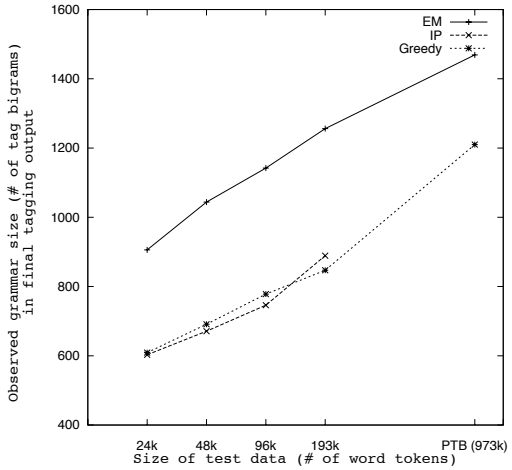


Figure 5: Comparison of observed grammar size (# of tag bigram types) in the final tagging output from EM, IP and MIN-GREEDY.

more than 3 hours without returning a solution.

It is interesting to see that for the 24k dataset, the greedy strategy finds a grammar set (containing only 478 tag bigrams). We observe that MIN-GREEDY produces 452 tag bigrams in the first minimization step (phase 1), and phase 2 adds another 26 entries, yielding a total of 478 tag bigrams in the final minimized grammar set. That is almost as good as the optimal solution (459 tag bigrams from IP) for the same problem. But MIN-GREEDY clearly has an advantage since it runs much faster than IP (as shown in Figure 4). Figure 5 shows a plot with the size of the observed grammar (i.e., number of tag bigram types in the final tagging output) versus the size of the test data for EM, IP and MIN-GREEDY methods. The figure shows that unlike EM, the other two approaches reduce the grammar size considerably and we observe the same trend even when scaling

Test set	Average Speedup	Optimality Ratio
24k test	2.7	0.96
48k test	1.7	0.98
96k test	4.3	0.98
193k test	7.1	0.93

Figure 6: *Average speedup* versus *Optimality ratio* computed for the model minimization step (when using MIN-GREEDY over IP) on different datasets.

to larger data. Minimizing the grammar size helps remove many spurious tag combinations from the grammar set, thereby yielding huge improvements in tagging accuracy over the EM method (Figure 3). We observe that for the 193k dataset, the final observed grammar size is greater for IP than MIN-GREEDY. This is because the alternating EM steps following the model minimization step add more tag bigrams to the grammar.

We compute the optimality ratio of the MIN-GREEDY approach with respect to the grammar size as follows:

$$\text{Optimality ratio} = \frac{\text{Size of IP grammar}}{\text{Size of MIN-GREEDY grammar}}$$

A value of 1 for this ratio implies that the solution found by MIN-GREEDY algorithm is exact. Figure 6 compares the optimality ratio versus average speedup (in terms of running time) achieved in the minimization step for the two approaches. The figure illustrates that our solution is nearly optimal for all data settings with significant speedup.

The MIN-GREEDY algorithm presented here can also be applied to scenarios where the dictionary is incomplete (i.e., entries for all word types are not present in the dictionary) and rare words

Method	Tagging accuracy (%)	Number of unique tag bigrams in final tagging output
EM	83.4	1195
IP	88.0	875
MIN-GREEDY	88.0	880

Figure 7: Results for unsupervised Italian POS tagging with a dictionary using a set of 90 tags.

take on all tag labels. In such cases, we can follow a similar approach as Ravi and Knight (2009) to assign tag possibilities to every unknown word using information from the known word/tag pairs present in the dictionary. Once the completed dictionary is available, we can use the procedure described in Section 5 to minimize the size of the grammar, followed by EM training.

6.2 Italian POS Tagging

We also compare the three approaches for Italian POS tagging and show results.

Data: We use the Italian CCG-TUT corpus (Bos et al., 2009), which contains 1837 sentences. It has three sections: newspaper texts, civil code texts and European law texts from the JRC-Acquis Multilingual Parallel Corpus. For our experiments, we use the POS-tagged data from the CCG-TUT corpus, which uses a set of 90 tags. We created a tag dictionary consisting of 8,733 word/tag pairs derived from the entire corpus (42,100 word tokens). We then created a test set consisting of 926 sentences (21,878 word tokens) from the original corpus. The per-token ambiguity for the test data is about 1.6 tags/token.

Results: Figure 7 shows the results on Italian POS tagging. We observe that MIN-GREEDY achieves significant improvements in tagging accuracy over the EM method and comparable to IP method. This also shows that the idea of model minimization is a general-purpose technique for such applications and provides good tagging accuracies on other languages as well.

7 Conclusion

We present a fast and efficient two-stage greedy minimization approach that can replace the integer programming step in (Ravi and Knight, 2009). The greedy approach finds close-to-optimal solutions for the minimization problem. Our algorithm runs much faster and achieves accuracies close to state-of-the-art. We also evaluate our method on test sets of varying sizes and show that our approach outperforms standard EM by a significant margin. For future work, we would like to incorporate some linguistic constraints within the greedy method. For example, we can assign higher costs to unlikely tag combinations (such as "SYM SYM", etc.).

Our greedy method can also be used for solving other unsupervised tasks where model minimization using integer programming has proven successful, such as word alignment (Bodrumlu et al., 2009).

Acknowledgments

The authors would like to thank Shang-Hua Teng and Anup Rao for their helpful comments and also the anonymous reviewers. This work was jointly supported by NSF grant IIS-0904684, DARPA contract HR0011-06-C-0022 under subcontract to BBN Technologies and DARPA contract HR0011-09-1-0028.

References

Bodrumlu, T., K. Knight, and S. Ravi. 2009. A new objective function for word alignment. In *Proceedings of the NAACL/HLT Workshop on Integer Programming for Natural Language Processing*.

Bos, J., C. Bosco, and A. Mazzei. 2009. Converting a dependency treebank to a categorial grammar treebank for Italian. In *Proceedings of the Eighth International Workshop on Treebanks and Linguistic Theories (TLT8)*.

Clarke, J. and M. Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research (JAIR)*, 31(4):399–429.

Dempster, A.P., N.M. Laird and D.B. Rubin. 1977. Maximum likelihood from incomplete data via the

EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38.

Garey, M. R. and D. S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. John Wiley & Sons.

Goldberg, Y., M. Adler, and M. Elhadad. 2008. EM can find pretty good HMM POS-taggers (when given a good start). In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL/HLT)*.

Goldwater, Sharon and Thomas L. Griffiths. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Martins, A., N. A. Smith, and E. P. Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics (ACL) and the 4th International Joint Conference on Natural Language Processing of the AFNLP*.

Merialdo, B. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2):155–171.

Punyakanok, V., D. Roth, W. Yih, and D. Zimak. 2004. Semantic role labeling via integer linear programming inference. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

Ravi, S. and K. Knight. 2008. Attacking decipherment problems optimally with low-order n-gram models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ravi, S. and K. Knight. 2009. Minimized models for unsupervised part-of-speech tagging. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics (ACL) and the 4th International Joint Conference on Natural Language Processing of the AFNLP*.

Smith, N. and J. Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Bringing Active Learning to Life

Ines Rehbein
Computational Linguistics
Saarland University

Josef Ruppenhofer
Computational Linguistics
Saarland University

Alexis Palmer
Computational Linguistics
Saarland University

{rehbein|josefr|apalmer}@coli.uni-sb.de

Abstract

Active learning has been applied to different NLP tasks, with the aim of limiting the amount of time and cost for human annotation. Most studies on active learning have only simulated the annotation scenario, using prelabelled gold standard data. We present the first active learning experiment for Word Sense Disambiguation with human annotators in a realistic environment, using fine-grained sense distinctions, and investigate whether AL can reduce annotation cost and boost classifier performance when applied to a real-world task.

1 Introduction

Active learning has recently attracted attention as having the potential to overcome the knowledge acquisition bottleneck by limiting the amount of human annotation needed to create training data for statistical classifiers. Active learning has been shown, for a number of different NLP tasks, to reduce the number of manually annotated instances needed for obtaining a consistent classifier performance (Hwa, 2004; Chen et al., 2006; Tomanek et al., 2007; Reichart et al., 2008).

The majority of such results have been achieved by simulating the annotation scenario using prelabelled gold standard annotations as a stand-in for real-time human annotation. Simulating annotation allows one to test different parameter settings without incurring the cost of human annotation. There is, however, a major drawback: we do not know whether the results of experiments performed using hand-corrected data carry over to real-world scenarios in which individual human annotators produce noisy annotations. In addition, we do not know to what extent error-prone annotations mislead the learning process. A systematic study of the impact of erroneous annotation on classifier performance in an active learning (AL) setting is overdue. We need to know a) whether the AL approach can really improve classifier performance and save annotation time when applied in a real-world scenario with noisy data, and b) whether AL works for classification tasks with fine-grained or complex annotation schemes and a low inter-annotator agreement.

In this paper we bring active learning to life in the context of frame semantic annotation of German texts within the SALSA project (Burchardt et al., 2006). Specifically, we apply AL methods for learning to assign semantic frames to predicates, following Erk (2005) in treating frame assignment as a Word Sense Disambiguation task. Under our fine-grained annotation scheme, annotators have to deal with a high level of ambiguity, resulting in low inter-annotator agreement for some word senses. This fact, along with the potential for wrong annotation decisions or possible biases from individual annotators, results in an annotation environment in which we get noisy data which might mislead the classifier. A second characteristic of our scenario is that there is no gold standard for the newly annotated data, which means that evaluation is not straightforward. Finally, we have multiple annotators whose deci-

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 949–957,
Beijing, August 2010

sions on particular instances may diverge, raising the question of which annotations should be used to guide the AL process. This paper thus investigates whether active learning can be successfully applied in a real-world scenario with the particular challenges described above.

Section 2 of the paper gives a short overview of the AL paradigm and some related work, and Section 3 discusses the multi-annotator scenario. In Section 4 we present our experimental design and describe the data we use. Section 5 presents results, and Section 6 concludes.

2 Active Learning

The active learning approach aims to reduce the amount of manual annotation needed to create training data sufficient for developing a classifier with a given performance. At each iteration of the AL cycle, the actual knowledge state of the learner guides the learning process by determining which instances are chosen next for annotation. The main goal is to advance the learning process by selecting instances which provide important information for the machine learner.

In a typical active learning scenario, a small set of manually labelled seed data serves as the initial training set for the classifier (learner). Based on the predictions of the classifier, a large pool of unannotated instances is queried for the next instance (or batch of instances) to be presented to the human annotator (sometimes called the oracle). The underlying active learning algorithm controlling the learning process tries to select the most informative instances in order to get a strong boost in classifier performance. Different methods can be used for determining informativity of instances. We use uncertainty sampling (Cohn et al., 1995) in which "most informative" instances are those for which the classifier has the lowest confidence in its label predictions. The rough intuition behind this selection method is that it identifies instance types which have yet to be encountered by the classifier. The learning process proceeds by presenting the selected instances to the human annotator, who assigns the correct label. The newly-annotated instances are added to the seed data and the classifier is re-trained on the new data set. The newly trained classifier now picks the next instances, based on its updated knowledge, and the process repeats. If the learning process can provide precisely that information which the classifier still needs to learn, a smaller number of instances should suffice to achieve the same accuracy as on a larger training set of randomly selected training examples.

Active learning has been applied to a number of natural language processing tasks like POS tagging (Ringger et al., 2007), NER (Laws and Schütze, 2008; Tomanek and Hahn, 2009), syntactic parsing (Osborne and Baldridge, 2004; Hwa, 2004), Word Sense Disambiguation (Chen et al., 2006; Chan and Ng, 2007; Zhu and Hovy, 2007; Zhu et al., 2008) and morpheme glossing for language documentation (Baldridge and Palmer, 2009). While most of these studies successfully show that the same classification accuracy can be achieved with a substantially smaller data set, these findings are mostly based on simulations using gold standard data.

For our task of Word Sense Disambiguation (WSD), mixed results have been achieved. AL seems to improve results in a WSD task with coarse-grained sense distinctions (Chan and Ng, 2007), but the results of (Dang, 2004) raise doubts as to whether AL can successfully be applied to a fine-grained annotation scheme, where Inter-Annotator Agreement (IAA) is low and thus the consistency of the human annotations decreases. In general, AL has been shown to reduce the cost of annotation when applied to classification tasks where a single human annotator predicts labels for new data points with a reasonable consistency and accuracy. It is not clear whether the same settings can be applied to a multi-annotator environment where IAA is low.

3 Active Learning in a realistic task including multiple annotators

Another possible difference between active learning simulations and real-world scenarios is the multi-annotator environment. In such a setting, two or more annotators assign labels to the same instances, which are then merged to check for conflicting decisions from different annotators. This is standard practise in many annotation projects doing fine-grained semantic annotation with a

high level of ambiguity, and it necessitates that all annotators work on the same data set.

Replicating an active learning simulation on hand-corrected data, starting with a fixed set of seed data and fixed parameter settings, using the same algorithm, will always result in the same training set selected from the pool. Human annotators, however, will assign different labels to the same instances, thus influencing the selection of the next instance from the pool. This means that individual annotators might end up with very different sets of annotated data, depending on factors like their interpretation of the annotation guidelines, an implicit bias towards a particular label, or simply errors made during annotation.

There is not much work addressing this problem. (Donmez and Carbonell, 2008) consider modifications of active learning to accommodate variability of annotators. (Baldridge and Palmer, 2009) present a real-world study with human annotators in the context of language documentation. The task consists of producing interlinear glossed text, including morphological and grammatical analysis, and can be described as a sequence labelling task. Annotation cost is measured as the actual time needed for annotation. Among other settings, the authors compare the performance of two annotators with different grades of expertise. The classifier trained on the data set created by the expert annotator in an active learning setting does obtain a higher accuracy on the gold standard. For the non-expert annotator, however, the active learning setting resulted in a lower accuracy than for a classifier trained on a randomly selected data set. This finding suggests that the quality of annotation needs to be high enough for active learning to actually work, and that annotation noise is a problem for AL.

There are two problems arising from this:

1. It is not clear whether active learning will work when applied to noisy data
2. It is not straightforward to apply active learning to a real-world scenario, where low IAA asks for multiple annotators

In our experiment we address these questions by systematically investigating the impact of annotation noise on classifier performance and on the composition of the training set. The next section presents the experimental design and the data used in our experiment.

4 Experimental Design

In the experiment we annotated 8 German causation nouns, namely *Ausgang, Anlass, Ergebnis, Resultat, Grund, Konsequenz, Motiv, Quelle* (outcome, occasion, effect, result, reason, consequence, motive, source of experience). These nouns were chosen because they exhibit a range of difficulty in terms of the number of senses they have in our annotation scheme. They all encode subtle distinctions between different word senses, but some of them are clearly easier to disambiguate than others. For instance, although *Ausgang* has 9 senses, they are easier to distinguish for humans than the 4 senses of *Konsequenz*.

Six annotators participated in the experiment. While all annotators were trained, having at least one year experience in frame-semantic annotation, one of the annotators is an expert with several years of training and working experience in the Berkeley FrameNet Project. This annotator also defined the frames (word senses) used in our experiment.

Prior to the experiment, all annotators were given 100 randomly chosen sentences. After annotating the training data, problematic cases were discussed to make sure that the annotators were familiar with the fine-grained distinctions between word senses in the annotation scheme. The data sets used for training were adjudicated by two of the annotators (one of them being the expert) and then used as a gold standard to test classifier performance in the active learning process.

4.1 Data and Setup

For each lemma we extracted sentences from the Wahrig corpus[1] containing this particular lemma. The annotators had to assign word senses to 300 instances for each target word, split into 6 packages of 50 sentences each. This resulted in 2,400 annotated instances per annotator (14,400 annotated instances in total). The annotation was done

[1]The Wahrig corpus includes more than 113 mio. sentences from German newspapers and magazines covering topics such as politics, science, fashion, and others.

Anlass	**Motiv**	**Konsequenz**	**Quelle**	**Ergebnis / Resultat**	**Ausgang**	**Grund**
Occasion (37)	Motif (47)	Causation (32)	Relational_nat_feat.(3)	Causation (4/10)	Outcome (67)	Causation (24)
Reason (63)	Reason(53)	Level_of_det.(6)	Source_of_getting (14)	Competitive_score(12/36)	Have_leave (4)	Reason (58)
		Response (61)	Source_of_exp. (14)	Decision (11/6)	Portal (21)	Death (1)
		MWE1 (1)	Source_of_info. (56)	Efficacy (2/3)	Outgoing_goods (4)	Part_orientation. (0)
			Well (6)	Finding_out (24/23)	Ostomy (0)	Locale_by_owner(3)
			Emissions_source (7)	Mathematics (1/0)	Origin (5)	Surface_earth (0)
				Operating_result (36/5)	Tech_output (7)	Bottom_layer (0)
				Outcome (10/17)	Process_end (2)	Soil (1)
					Departing (1)	CXN1 (0)
						CXN2 (0)
						MWE1 (0)
						MWE2 (10)
						MWE3 (0)
						MWE4 (3)
						MWE5 (0)
						MWE6 (0)
Fleiss' kappa for the 6 annotators for the 150 instances annotated in the random setting						
0.67	0.79	0.55	0.77	0.63 / 0.59	0.82	0.43

Table 1: 8 causation nouns and their word senses (numbers in brackets give the distribution of word senses in the gold standard (100 sentences); CXN: constructions, MWE: multi-word expressions; note that Ergebnis and Resultat are synonyms and therefore share the same set of frames.)

using a Graphical User Interface where the sentence was presented to the annotator, who could choose between all possible word senses listed in the GUI. The annotators could either select the frame by mouse click or use keyboard shortcuts. For each instance we recorded the time it took the annotator to assign an appropriate label. To ease the reading process the target word was highlighted.

As we want to compare time requirements needed for annotating random samples and sentences selected by active learning, we had to control for training effects which might speed up the annotation. Therefore we changed the annotation setting after each package, meaning that the first annotator started with 50 sentences randomly selected from the pool, then annotated 50 sentences selected by AL, followed by another 50 randomly chosen sentences, and so on. We divided the annotators into two groups of three annotators each. The first group started annotating in the random setting, the second group in the AL setting. The composition of the groups was changed for each lemma, so that each annotator experienced all different settings during the annotation process. The annotators were not aware of which setting they were in.

Pool data For the random setting we randomly selected three sets of sentences from the Wahrig corpus which were presented for annotation to all six annotators. This allows us to compare annotation time and inter-annotator agreement between the annotators. For the active learning setting we randomly selected three sets of 2000 sentences each, from which the classifier could pick new instances during the annotation process. This means that for each trial the algorithm could select 50 instances out of a pool of 2000 sentences. On any given AL trial each annotator uses the same pool as all the other annotators. In an AL simulation with fixed settings and gold standard labels this would result in the same subset of sentences selected by the classifier. For our human annotators, however, due to different annotation decisions the resulting set of sentences is expected to differ.

Sampling method Uncertainty sampling is a standard sampling method for AL where new instances are selected based on the confidence of the classifier for predicting the appropriate label. During early stages of the learning process when the classifier is trained on a very small seed data set, it is not beneficial to add the instances with the lowest classifier confidence. Instead, we use a dynamic version of uncertainty sampling (Rehbein and Ruppenhofer, 2010), based on the confidence of a maximum entropy classifier[2], taking into account how much the classifier has learned so far. In each iteration one new instance is selected from the pool and presented to the oracle. After annotation the classifier is retrained on the new data set. The modified uncertainty sampling results in a more robust classifier performance during early stages of the learning process.

[2] `http://maxent.sourceforge.net`

	Anlass		Motiv		Konsequenz		Quelle		Ergebnis		Resultat		Ausgang		Grund	
	R	U	R	U	R	U	R	U	R	U	R	U	R	U	R	U
A_1	8.6	9.6	5.9	6.6	10.7	10.5	6.0	4.8	10.5	7.4	10.1	9.6	6.4	10.0	10.2	11.1
A_2	4.4	5.7	4.8	5.9	8.2	9.2	4.9	4.9	6.4	4.4	11.7	8.5	5.1	7.7	9.0	9.3
A_3	9.9	9.2	6.8	6.7	6.8	8.3	7.4	6.1	9.4	7.6	9.0	12.3	7.5	8.5	11.7	10.2
A_4	5.8	4.9	3.6	3.6	9.9	11.3	4.8	3.5	7.9	7.1	9.7	11.1	3.6	4.1	9.9	9.4
A_5	3.0	3.5	3.0	2.6	4.8	4.9	3.8	3.0	6.8	4.8	6.7	6.1	3.1	3.5	6.3	6.0
A_6	5.4	6.3	5.3	4.7	6.7	8.6	5.4	4.6	7.8	6.1	8.7	9.0	6.9	6.6	9.3	8.5
ø	**6.2**	**6.5**	**4.9**	**5.0**	**7.8**	**8.8**	**5.4**	**4.5**	**8.1**	**6.2**	**9.3**	**9.4**	**5.4**	**6.7**	**9.4**	**9.1**
sl	25.8	27.8	27.8	26.0	24.2	25.8	24.9	26.5	25.7	25.2	29.0	35.9	25.5	27.9	26.8	29.7

Table 2: Annotation time (sec/instance) per target/annotator/setting and average sentence length (sl)

5 Results

The basic idea behind active learning is to select the most informative instances for annotation. The intuition behind "more informative" is that these instances support the learning process, so we might need fewer annotated instances to achieve a comparable classifier performance, which could decrease the cost of annotation. On the other hand, "more informative" also means that these instances might be more difficult to annotate, so it is only fair to assume that they might need more time for annotation, which increases annotation cost. To answer the question of whether AL reduces annotation cost or not we have to check a) how long it took the annotators to assign labels to the AL samples compared to the randomly selected instances, and b) how many instances we need to achieve the best (or a sufficient) performance in each setting. Furthermore, we want to investigate the impact of active learning on the distribution of the resulting training sets and study the correlation between the performance of the classifier trained on the annotated data and these factors: the difficulty of the annotation task (assessed by IAA), expertise and individual properties of the annotators.

5.1 Does AL speed up the annotation process when working with noisy data?

Table 2 reports annotation times for each annotator and target for random sampling (R) and uncertainty sampling (U). For 5 out of 8 targets the time needed for annotating in the AL setting (averaged over all annotators) was higher than for annotating the random samples. To investigate whether this might be due to the length of the sentences in the samples, Table 2 shows the average sentence length for random samples and AL samples for each target lemma. Overall, the sentences selected by the classifier during AL are longer (26.2 vs. 28.1 token per sentence), and thus may take the annotators more time to read.[3] However, we could not find a significant correlation (Spearman rank correlation test) between sentence length and annotation time, nor between sentence length and classifier confidence.

The three target lemmas which took longer to annotate in the random setting are *Ergebnis* (result), *Grund* (reason) and *Quelle* (source of experience). This observation cannot be explained by sentence length. While sentence length for *Ergebnis* is nearly the same in both settings, for *Grund* and *Quelle* the sentences picked by the classifier in the AL setting are significantly longer and therefore should have taken more time to annotate. To understand the underlying reason for this we have to take a closer look at the distribution of word senses in the data.

5.2 Distribution of word senses in the data

In the literature it has been stated that AL implicitly alleviates the class imbalance problem by extracting more balanced data sets, while random sampling tends to preserve the sense distribution present in the data (Ertekin et al., 2007). We could not replicate this finding when using noisy data to guide the learning process. Table 3 shows the distribution of word senses for the target lemma *Ergebnis* a) in the gold standard, b) in the random samples, and c) in the AL samples.

The variance in the distribution of word senses in the random samples and the gold standard can

[3]The correlation between sentence length and annotation time is not obvious, as the annotators only have to label one target in each sentence. For ambiguous sentences, however, reading time may be longer, while for the clear cases we do not expect a strong effect.

Ergebnis Frame	gold (%)	R (%)	U (%)
Causation	4.0	4.8	3.7
Outcome	10.0	17.8	10.5
Finding_out	24.0	26.2	8.2
Efficacy	2.0	0.8	0.1
Decision	11.0	5.1	3.2
Mathematics	1.0	1.6	0.4
Operating_result	36.0	24.5	66.7
Competitive_score	12.0	19.2	7.2

Table 3: Distribution of frames (word senses) for the lemma *Ergebnis* in the gold standard (100 sentences), in the random samples (R) and AL samples (U) (150 sentences each)

	Random			Uncertainty		
	50	100	150	50	100	150
Anlass	0.85	**0.86**	0.85	0.84	0.85	0.84
Motiv	0.57	0.62	0.63	0.64	0.67	**0.70**
Konseq.	0.55	0.59	0.60	0.61	**0.62**	**0.62**
Quelle	0.56	0.53	0.54	0.52	0.52	**0.57**
Ergebnis	0.39	**0.42**	0.41	0.39	0.37	0.38
Resultat	0.31	0.35	**0.37**	0.32	0.34	0.34
Ausgang	0.67	0.69	0.69	0.68	0.72	**0.74**
Grund	**0.48**	0.47	0.47	0.47	0.44	**0.48**

Table 4: Avg. classifier performance (acc.) over all annotators for the 8 target lemmas when training on 50, 100 and 150 annotated instances for random samples and uncertainty samples

be explained by low inter-annotator agreement caused by the high level of ambiguity for the target lemmas. The frame distribution in the data selected by uncertainty sampling, however, crucially deviates from those of the gold standard and the random samples. A disproportionately high 66% of the instances selected by the classifier have been assigned the label Operating_result by the human annotators. This is the more surprising as this frame is fairly easy for humans to distinguish.

The classifier, however, proved to have serious problems learning this particular word sense and thus repeatedly selected more instances of this frame for annotation. As a result, the distribution of word senses in the training set for the uncertainty samples is highly skewed, having a negative effect on the overall classifier performance. The high percentage of instances of the "easy-to-decide" frame Operating_result explains why the instances for *Ergebnis* took less time to annotate in the AL setting. Thus we can conclude that annotating the same number of instances on average takes more time in the AL setting, and that this effect is not due to sentence length.

5.3 What works, what doesn't, and why

For half of the target lemmas *(Motiv, Konsequenz, Quelle, Ausgang)*, we did obtain best results in the AL setting (Table 4). For *Ausgang* and *Motiv* AL gives a substantial boost in classifier performance of 5% and 7% accuracy, while the gains for *Konsequenz* and *Quelle* are somewhat smaller with 2% and 1%, and for *Grund* the highest accuracy was reached on both the AL and the random sample.

Figure 1 (top row) shows the learning curves for *Resultat*, our worst-performing lemma, for the classifier trained on the manually annotated samples for each individual annotator. The solid black line represents the majority baseline, obtained by assigning the most frequent word sense in the gold standard to all instances. For both random and AL settings, results are only slightly above the baseline. The curves for the AL setting show how erroneous decisions can mislead the classifier, resulting in classifier accuracy below the baseline for two of the annotators, while the learning curves for these two annotators on the random samples show the same trend as for the other 4 annotators.

For *Konsequenz* (Figure 1, middle), the classifier trained on the AL samples yields results over the baseline after around 25 iterations, while in the random sampling setting it takes at least 100 iterations to beat the baseline. For *Motiv* (Figure 1, bottom row), again we observe far higher results in the AL setting. A possible explanation for why AL seems to work for *Ausgang, Motiv* and *Quelle* might be the higher IAA[4] (κ 0.825, 0.789, 0.768) as compared to the other target lemmas. This, however, does not explain the good results achieved on the AL samples for *Konsequenz*, for which IAA was quite low with κ 0.554.

Also startling is the fact that AL seems to work particularly well for one of the annotators (A_6, Figure 1) but not for others. Different possible explanations come to mind: (a) the accuracy of the annotations for this particular annotator, (b) the

[4]IAA was computed on the random samples, as the AL samples do not include the same instances.

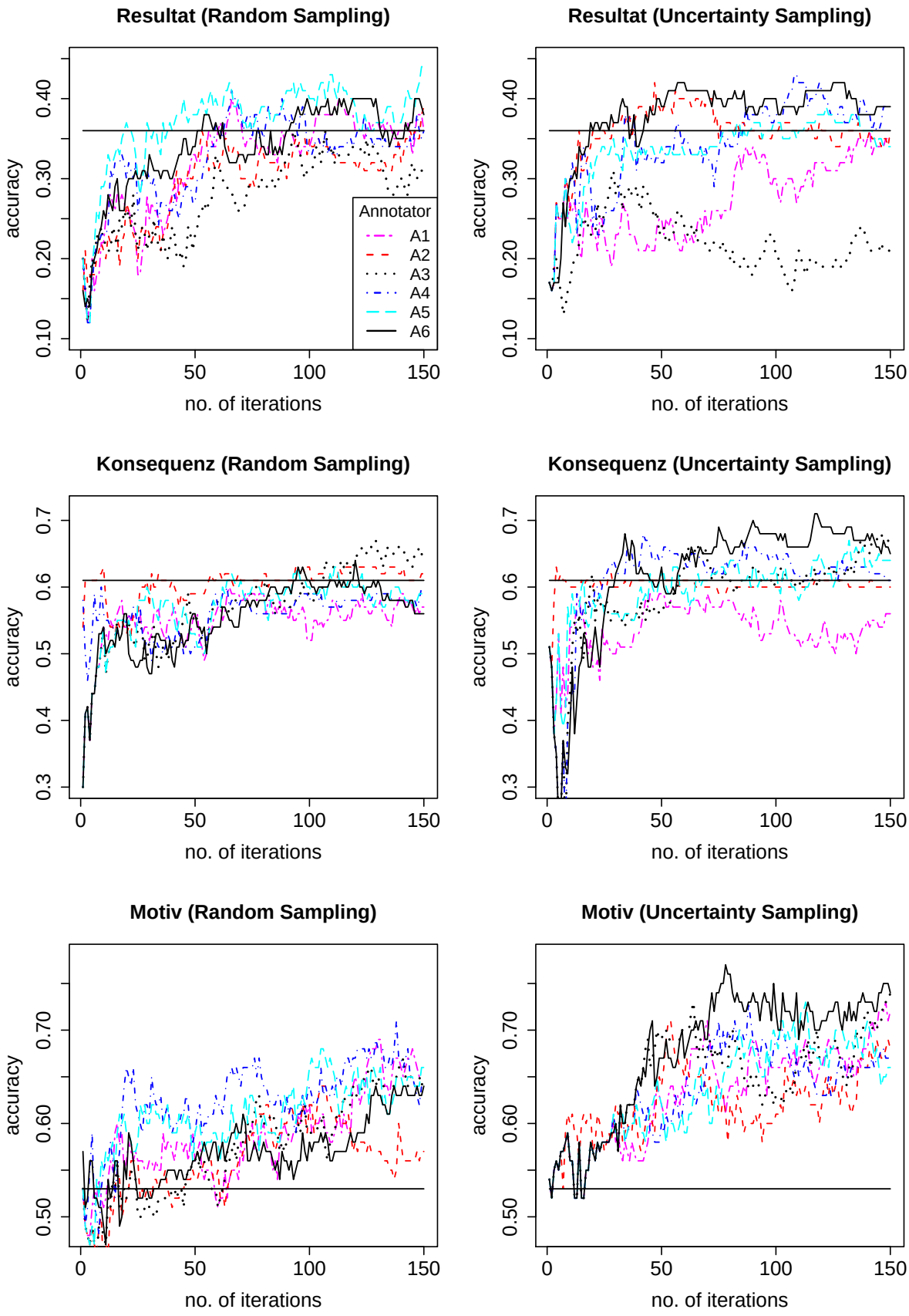


Figure 1: Active learning curves for Resultat, Konsequenz and Motiv (random sampling versus uncertainty sampling; the straight black line shows the majority baseline)

Konsequenz	A1	A2	A3	A4	A5	A6
human	0.80	0.72	0.89	0.73	0.89	0.76
maxent	0.60	0.63	0.67	0.60	0.63	0.64

Table 5: Acc. for human annotators against the adjudicated random samples and for the classifier

instances selected by the classifier based on the annotation decisions of the individual annotators, and (c) the distribution of frames in the annotated training sets for the different annotators.

To test (a) we evaluated the annotated random samples for *Konsequenz* for each annotator against the adjudicated gold standard. Results showed that there is no strong correlation between the accuracy of the human annotations and the performance of the classifier trained on these annotations. The annotator for whom AL worked best had a medium score of 0.76 only, while the annotator whose annotations were least helpful for the classifier showed a good accuracy of 0.80 against the gold standard.

Next we tested (b) the impact of the particular instances in the AL samples for the individual annotators on classifier performance. We took all instances in the AL data set from A_6, whose annotations gave the greatest boost to the classifier, removed the frame labels and gave them to the remaining annotators for re-annotation. Then we trained the classifier on each of the re-annotated samples and compared classifier performance. Results for 3 of the remaining annotators were in the same range or even higher than the ones for A_6 (Figure 2). For 2 annotators, however, results remained far below the baseline.

This again shows that the AL effect is not directly dependent on the accuracy of the individual annotators, but that particular instances are more informative for the classifier than others. Another crucial point is (c) the distribution of frames in the samples. In the annotated samples for A_1 and A_2 the majority frame for *Konsequenz* is Causation, while in the samples for the other annotators Response was more frequent. In our test set Response also is the most frequent frame, therefore it is not surprising that the classifiers trained on the samples of A_3 to A_6 show a higher performance. This means that high-quality annotations (identified by IAA) do not necessarily provide the information from which the classifier benefits most, and that in a realistic annotation task addressing the class imbalance problem (Zhu and Hovy, 2007) is crucial.

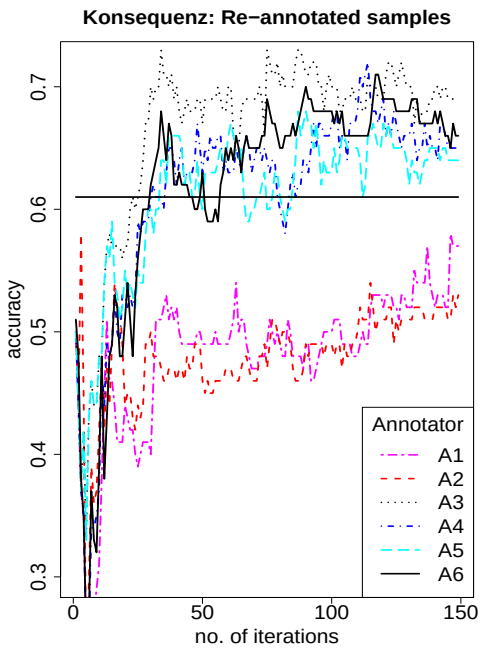


Figure 2: Re-annotated instances for Konsequenz (AL samples from annotator A_6)

6 Conclusions

We presented the first experiment applying AL in a real-world scenario by integrating the approach in an ongoing annotation project. The task and annotation environment pose specific challenges to the AL paradigm. We showed that annotation noise caused by biased annotators as well as erroneous annotations mislead the classifier and result in skewed data sets, and that for this particular task no time savings are to be expected when applied to a realistic scenario. Under certain conditions, however, classifier performance can improve over the random sampling baseline even on noisy data and thus yield higher accuracy in the active learning setting. Critical features which seem to influcence the outcome of AL are the amount of noise in the data as well as the distribution of frames in training- and test sets. Therefore, addressing the class imbalance problem is crucial for applying AL to a real annotation task.

Acknowledgments

This work was funded by the German Research Foundation DFG (grant PI 154/9-3 and the MMCI Cluster of Excellence).

References

Baldridge, Jason and Alexis Palmer. 2009. How well does active learning actually work?: Time-based evaluation of cost-reduction strategies for language documentation. In *Proceedings of EMNLP 2009*.

Burchardt, Aljoscha, Katrin Erk, Anette Frank, Andrea Kowalski, Sebastian Padó, and Manfred Pinkal. 2006. The salsa corpus: a german corpus resource for lexical semantics. In *Proceedings of LREC-2006*.

Chan, Yee Seng and Hwee Tou Ng. 2007. Domain adaptation with active learning for word sense disambiguation. In *Proceedings of ACL-2007*.

Chen, Jinying, Andrew Schein, Lyle Ungar, and Martha Palmer. 2006. An empirical study of the behavior of active learning for word sense disambiguation. In *Proceedings of NAACL-2006*, New York, NY.

Cohn, David A., Zoubin Ghahramani, and Michael I. Jordan. 1995. Active learning with statistical models. In Tesauro, G., D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 705–712. The MIT Press.

Dang, Hoa Trang. 2004. *Investigations into the role of lexical semantics in word sense disambiguation*. PhD dissertation, University of Pennsylvania, Pennsylvania, PA.

Donmez, Pinar and Jaime G. Carbonell. 2008. Proactive learning: Cost-sensitive active learning with multiple imperfect oracles. In *Proceedings of CIKM08*.

Erk, Katrin. 2005. Frame assignment as word sense disambiguation. In *Proceedings of the IWCS-6*.

Ertekin, Şeyda, Jian Huang, L'eon Bottou, and Lee Giles. 2007. Learning on the border: active learning in imbalanced data classification. In *Proceedings of CIKM '07*.

Hwa, Rebecca. 2004. Sample selection for statistical parsing. *Computational Linguistics*, 30(3):253–276.

Laws, Florian and Heinrich Schütze. 2008. Stopping criteria for active learning of named entity recognition. In *Proceedings of Coling 2008*.

Osborne, Miles and Jason Baldridge. 2004. Ensemble-based active learning for parse selection. In *Proceedings of HLT-NAACL 2004*.

Rehbein, Ines and Josef Ruppenhofer. 2010. Theres no data like more data? revisiting the impact of data size on a classification task. In *Proceedings of LREC-07, 2010*.

Reichart, Roi, Katrin Tomanek, Udo Hahn, and Ari Rappoport. 2008. Multi-task active learning for linguistic annotations. In *Proceedings of ACL-08: HLT*.

Ringger, Eric, Peter Mcclanahan, Robbie Haertel, George Busby, Marc Carmen, James Carroll, and Deryle Lonsdale. 2007. Active learning for part-of-speech tagging: Accelerating corpus annotation. In *Proceedings of ACL Linguistic Annotation Workshop*.

Tomanek, Katrin and Udo Hahn. 2009. Reducing class imbalance during active learning for named entity annotation. In *Proceedings of the 5th International Conference on Knowledge Capture*, Redondo Beach, CA.

Tomanek, Katrin, Joachim Wermter, and Udo Hahn. 2007. An approach to text corpus construction which cuts annotation costs and maintains corpus reusability of annotated data. In *Proceedings of EMNLP-CoNLL 2007*.

Zhu, Jingbo and Ed Hovy. 2007. Active learning for word sense disambiguation with methods for addressing the class imbalance problem. In *Proceedings of EMNLP-CoNLL 2007*.

Zhu, Jingbo, Huizhen Wang, Tianshun Yao, and Benjamin K. Tsou. 2008. Active learning with sampling by uncertainty and density for word sense disambiguation and text classification. In *Proceedings of Coling 2008*.

Computing EM-based Alignments of Routes and Route Directions as a Basis for Natural Language Generation

Michael Roth and **Anette Frank**
Department of Computational Linguistics
Heidelberg University
{mroth,frank}@cl.uni-heidelberg.de

Abstract

Route directions are natural language (NL) statements that specify, for a given navigational task and an automatically computed route representation, a sequence of actions to be followed by the user to reach his or her goal. A corpus-based approach to generate route directions involves (i) the selection of elements along the route that need to be mentioned, and (ii) the induction of a mapping from route elements to linguistic structures that can be used as a basis for NL generation.

This paper presents an Expectation-Maximization (EM) based algorithm that aligns geographical route representations with semantically annotated NL directions, as a basis for the above tasks. We formulate one basic and two extended models, the latter capturing special properties of the route direction task. Although our current data set is small, both extended models achieve better results than the simple model and a random baseline. The best results are achieved by a combination of both extensions, which outperform the random baseline and the simple model by more than an order of magnitude.

1 Introduction

The purpose of route directions is to inform a person, who is typically not familiar with his current environment, of how to get to a designated goal. Generating such directions poses difficulties on various conceptual levels such as planning the route, selecting landmarks (i.e., recognizable buildings or structures) and splitting the task into appropriate single instructions of how to navigate along the route using the selected landmarks as reference points.

Previously developed natural language generation (NLG) systems make use of simple heuristics for the task of content selection for route directions (Dale et al., 2005; Roth and Frank, 2009). In our work, we aim for a corpus-based approach that can be flexibly modeled after natural, human-produced directions for varying subtasks (e.g., indoor vs. outdoor navigation), and that facilitates multilingual extensions. By employing salient landmarks and allowing for variation in NL realization, such a system is expected to generate natural sounding directions that are easier to memorize and easier to follow than directions given by a classical route planner or navigation system.

NLG for route directions crucially differs from other generation tasks such as document summarization (Mani, 2001) in that the selection and ordering of input structures for language generation is heavily situation-dependent, i.e., dependent on the specific properties of a given route to be followed.

In line with a corpus-based NLG approach, we propose to automatically align geographical route representations as produced by a route planner with an annotated corpus of NL directions given by humans for the respective routes. The induced alignments will (i) serve to identify which elements of a route to select for verbalization, and (ii) deliver correspondences between route segments and linguistic input structures that can be used as a basis for statistical NL generation. We investi-

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 958–966,
Beijing, August 2010

gate a minimally supervised method for inducing such alignments to ensure maximal flexibility for adaptations to different scenarios.

The remainder of this paper is structured as follows: In Section 2 we discuss related work. Section 3 introduces the task, and the representation formats and resources we use. Section 4 introduces a basic Expectation-Maximization model and two extensions for the alignment task. Section 5 outlines the experiments and presents the evaluation results. In Section 6 we conclude and discuss future work.

2 Related Work

Various aspects of route directions have been subject of research in computational linguistics, ranging from instructional dialogues in MapTask (Anderson et al., 1991) to recent work on learning to follow route directions (Vogel and Jurafsky, 2010). However, little work has been done on generating NL directions based on data from Geographical Information Systems (Dale et al., 2005; Roth and Frank, 2009).

NLG systems are typically realized as pipeline architectures (Reiter and Dale, 2000). As a first step, they compute a set of messages that represent the information to be conveyed to a user, given a specific communicative task (*Content Selection*). Selecting appropriate content for a task can be defined heuristically, by manually crafted rules or by learning content selection rules automatically from corpus data. Previous work by Dale et al. (2005) and Roth and Frank (2009) on generating NL directions used hand-crafted heuristics. Duboue and McKeown (2003) were the first to model content selection as a machine learning task, in which selection rules are induced from pairs of human-written text and associated sets of database entries. They induce baseline selection rules from exact matches of NL expressions with database entries; in addition, class-based rules are computed by matching database entry types against NL expressions, using statistical co-occurrence clusters. Barzilay and Lapata (2005) incorporate the interplay between multiple events and entities when learning content selection rules using a special link function.

Recent work by Liang et al. (2009) focuses on modeling grounded language, by aligning real-world representations with NL text that references corresponding world states. They show how a generative model can be used to segment text into utterances and to identify relevant facts with minimal supervision. Both tasks are handled jointly in a unified framework by training a hierarchical semi-Markov model on pairs of text and world states, thereby modeling sequencing effects in the presentation of facts. While their work is not primarily concerned with NLG, the learned correspondences and their probabilities could be applied to induce content selection rules and linguistic mappings in a NLG task. The approach is shown to be effective in scenarios typical for NLG settings (weather forecasts, RoboCup sportscasting, NFL recaps) that differ in the amount of available data, length of textual descriptions, and density of alignments.

In the following, we will adapt ideas from their EM-based approach to align (segments of) route representations and NL route directions in a minimally supervised manner. We will investigate increasingly refined models that are tailored to the nature of our task and underlying representations. In particular, we extend their approach by exploiting semantic markup in the NL direction corpus.

3 Aligning Routes and Directions

In this work we explore the possibility of using an implementation of the EM algorithm (Dempster et al., 1977) to learn correspondences between (segments of) the geographical representation of a route and linguistic instructions of how to follow this route in order to arrive at a designated goal. We are specifically interested in identifying which parts of a route are realized in natural language and which kinds of semantic constructions are used to express them.

As a data source for inducing such correspondences we use a parallel corpus of route representations and corresponding route directions that were collected in a controlled experiment for navigation in an urban street network (cf. Schuldes et al. (2009)). For the alignment task, the routes were compiled to a specification format that has been realized in an internal version of an online route planner. Figure 1 displays the route rep-

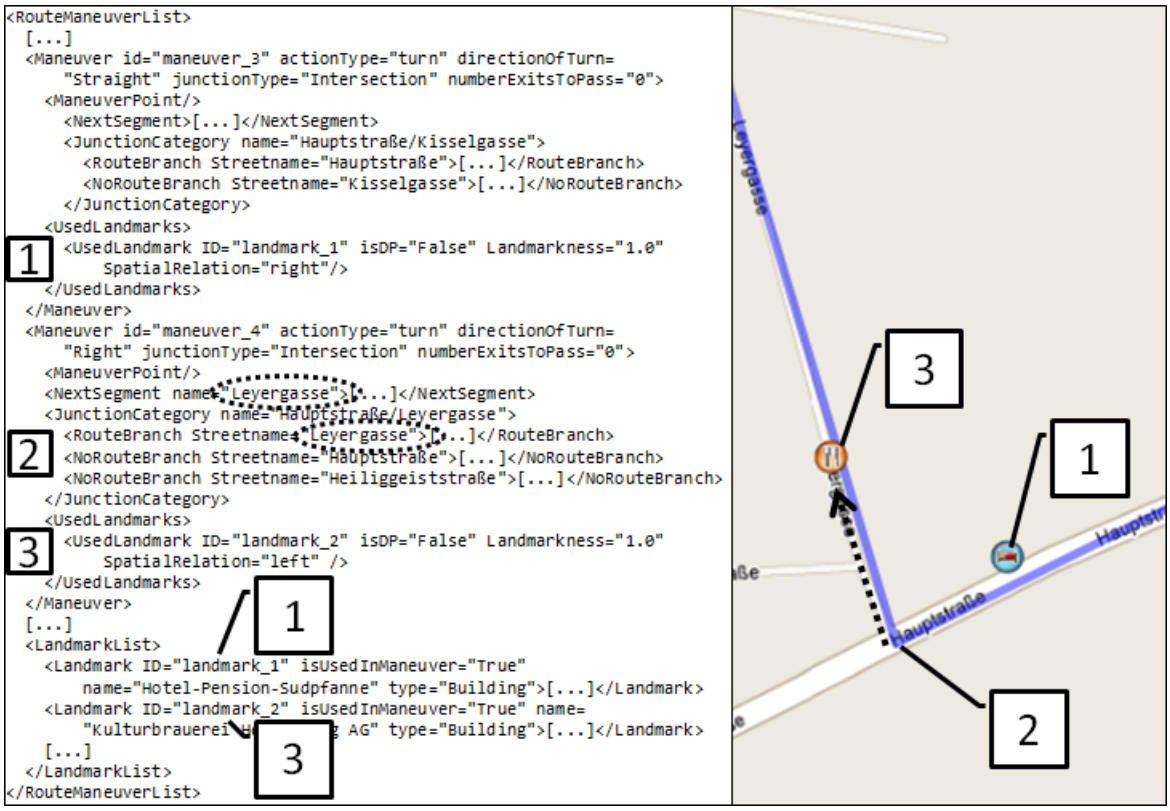


Figure 1: A (partial) route representation of the route segment displayed on the right.

resentation for a small route segment (a junction connecting 'Hauptstraße' and 'Leyergasse'). The corresponding part of a NL route direction is displayed in Figure 2. The route representation and the NL direction share some common concepts: For example, both contain references to a landmark called "Sudpfanne" (marked as [1]) and a street named "Leyergasse" (marked as [2]). Using pairs of route representations and directions, we aim to automatically induce alignments between such correspondences. In the following we describe our data in more detail.

3.1 Route Representation Format

The route representation format we use (illustrated in Figure 1) is an extended version of the *OpenGIS Location Service (OpenLS) Implementation Standards*, a set of XML-based representations specified by the Open Geospatial Consortium[1]. Previous approaches on extending the latter with landmarks in an interoperable way have been presented by Neis and Zipf (2008). The representation format of our data has been developed in close collaboration with researchers from Geoinformatics at Heidelberg University[2] and adopts ideas previously proposed in the *Cognitive OpenLS* specification by Hansen et al. (2006). The resulting specification will be implemented in an extended (internal) version of the online route planner OpenRouteService.org.

Our work revolves around two kinds of elements in this format: so-called *maneuvers*, i.e., elements that describe a decision point including the required action and the following route segment, and *landmarks* that occur along the route. For the alignment task we focus on the following types of attributes that are part of the XML specification, specified here as **Attribute (Element)**:

directionOfTurn (Maneuver) – the direction of movement for the current maneuver, i.e., "left", "right" or "straight"

[1]http://www.opengeospatial.org/standards/is

[2]Chair of GIScience, Alexander Zipf, http://www.geog.uni-heidelberg.de/lehrstuehle/gis/

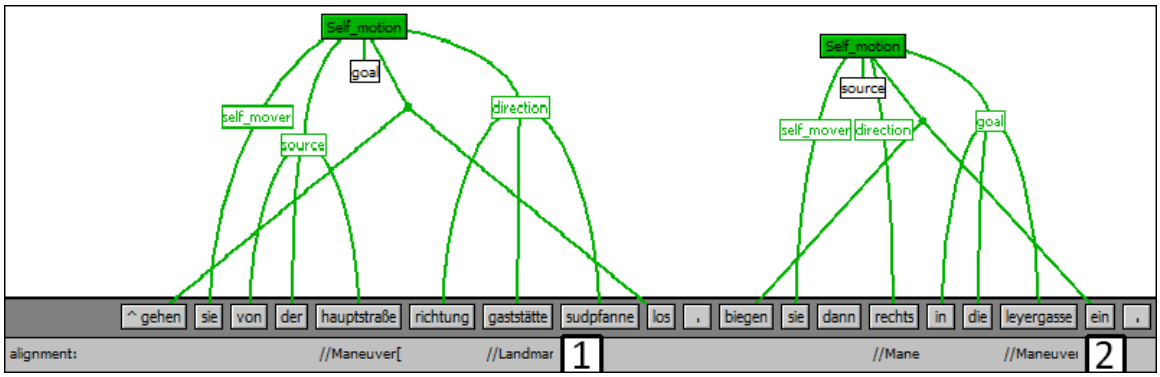


Figure 2: Directions for the route segment displayed in Figure 1 annotated with frame-semantic markup and alignment information. The directions translate to "You start walking from Hauptstraße towards Gaststätte Sudpfanne, then you turn right onto Leyergasse"

junctionType (Maneuver) – the type of junction at the current maneuver, e.g., "intersection", "crossing"

name (JunctionCategory) – the name of the junction at the current maneuver, e.g., "Hauptstraße/Leyergasse"

name (NextSegment) – the name of the street of the next route segment, e.g., "Hauptstraße"

streetName (RouteBranch) – the street name of a branch along which the route continues, e.g., "Leyergasse"

streetName (NoRouteBranch) – the street name of a branch that is not part of the route, e.g., "Kisselgasse"

name (Landmark) – the name of a landmark, e.g., "Hotel Sudpfanne"

spatialRelation (UsedLandmark) – the spatial relation between a landmark and the current maneuver, e.g., "left", "right", "before"

3.2 A Parallel Corpus of Route Directions

The corpus of route directions used in this work is a subset of the data collected by Schuldes et al. (2009) in a desk-based experiment. To elicit NL route directions, subjects were shown a web application that guided them along a route by means of a 2D animation. Subsequently they had to write NL route directions in German for the shown routes. The subjects were allowed to use all information displayed by the web application: named places, buildings, bridges and street names, etc. The resulting directions were POS-tagged with TreeTagger (Schmid, 1997), dependency-parsed with XLE (Maxwell and Kaplan, 1993), and manually revised. Additionally, we annotated frame-semantic markup (Fillmore et al., 2003) and gold standard alignments to the route representation using the SALTO annotation tool (Burchardt et al., 2006).

Frame semantic markup. The texts are annotated with an inventory of 4 frames relevant for directions (SELF_MOTION, PERCEPTION, BEING_LOCATED, LOCATIVE_RELATION), with semantic roles (*frame elements*) such as DIRECTION, GOAL, PATH, LOCATION. Figure 2 illustrates a typical example for the use of the SELF_MOTION frame, once with the elements SOURCE and DIRECTION, and once with the elements DIRECTION and GOAL. Our alignment model uses the frame semantic annotation as structuring information.

Gold standard alignments. For evaluation we constructed gold alignments. We asked two annotators to align text parts with corresponding attributes in the respective route representation[3]. The information about corresponding attributes was added to a single word by manually insert-

[3]The alignments have not been double annotated, hence no measure for inter-annotator agreement can be provided.

	#S	#W	#FE	#aligned FE
avg. per direction	8	98	28	14 (50%)
overall	412	5298	1519	750

Table 1: Corpus statistics: number of sentences (S), words (W), frame elements (FE) and alignments.

	#attributes	#aligned attr.
avg. per route	115	14 (12%)
overall	921	

Table 2: Corpus statistics: total number and percentage of relevant attribute alignments.

ing XPATH expressions that unambiguously refer to the aligned attribute in the route representation format. For learning the alignment model, the annotations were spread to all words in the span of the respective frame element.

Corpus statistics. We made use of a corpus of 54 NL directions collected for 8 routes in an urban street network. Tables 1 and 2 give some statistics about the number of words (W) and frame elements (FE) in the parallel corpus. Comparing the total number of relevant attributes (as listed in Section 3.1) and attributes annotated in the gold alignments (aligned attr.) we note that only 12% are actually mentioned in NL directions. Thus it is necessary to select the most salient attributes to avoid the generation of overly redundant text.

4 Alignment Model

For the induction of alignments between (parts of) route structures and semantic representations, we adopt ideas from the models presented in Liang et al. (2009) (cf. Section 2).

We start from a basic *frame alignment model*. It specifies a conditional probability distribution $\mathrm{p}(f|a)$ for the alignment to a frame element f of type f_t (e.g., source, goal, direction) in the frame-semantic annotation layer given an attribute a of type a_t (e.g., streetName, directionOfTurn) in the route representation format. Note that this model does not take into account the actual value a_v of the attribute a nor the words that are annotated as part of f. We assume that the frame annotation represents a reliable segmentation for this alignment. This allows us to omit modeling segmentation explicitly.

As extensions to the basic frame alignment model, we specify two further models that capture properties that are specific to the task of direction alignment. As route directions are typically presented in a linear order with respect to the route, we incorporate an additional *distance model* λ in our alignment. We further account for *word choice* within a frame element as an additional factor. The word choice model $\mathrm{p}(w|a)$ will exploit attribute type and value information in the route representations that are reflected in word choice in the linguistic instructions. Both extensions are inspired by and share similarities with models that have been successfully applied in work on text alignment for the task of machine translation (Vogel et al., 1996; Tiedemann, 2003).

Our full model is a distribution over frame elements f and words w that factorizes the three above mentioned parts under the assumption of independence between each component and each attribute:

$$\mathrm{p}(f, w|a) = \mathrm{p}(f|a)\lambda(\mathrm{dist}(f, a))\,\mathrm{p}(w|a) \quad (1)$$

The individual models are described in more detail in the following subsections.

4.1 Frame Alignment Model

This basic frame alignment model specifies the probabilities $\mathrm{p}(f|a)$ for aligning an attribute a of type a_t (i.e., one of the types listed in Section 3.1) to a frame element f labeled as type f_t. This alignment model is initialized as a uniform distribution over f and trained using a straight-forward implementation of the EM algorithm, following the well-known IBM Model 1 for alignment in machine translation (Brown et al., 1993). The expectation step (E-step) computes expected counts given occurrences of f_t and a_t under the assumption that all alignments are independent 1:1 correspondences:

$$\mathrm{count}(f_t, a_t) = \frac{\sum_{\{\langle f', a'\rangle | f'_t = f_t \wedge a'_t = a_t\}} \mathrm{p}(f'|a')}{\sum_{\{\langle f', y\rangle | f'_t = f_t\}} \mathrm{p}(f'|y)} \quad (2)$$

The probabilities are re-estimated to maximize the overall alignment probability by normalizing

the estimated counts (M-step):

$$\mathrm{p}(f|a) = \frac{\mathrm{count}(f_t, a_t)}{\sum_x \mathrm{count}(x_t, a_t)} \quad (3)$$

4.2 Distance Model

We hypothesize that the order of route directions tends to be consistent with the order of maneuvers encoded by the route representation. We include this information in our alignment model by defining a distance measure $\mathrm{dist}(f, a)$ between the relative position of a frame element f in the text and the relative position of an attribute a in the route representation. The probabilities are specified in form of a distance distribution $\lambda(i)$ over normalized distances $i \in [0:1]$ and learned during EM training. The weights are initialized as a uniform distribution and re-estimated in each M-step by normalizing the estimated counts:

$$\lambda(i) = \frac{\sum_{\{\langle x,y\rangle |\, \mathrm{dist}(x,y)=i\}} \mathrm{count}(x, y)}{\sum_{\{\langle x,y\rangle\}} \mathrm{count}(x, y)} \quad (4)$$

4.3 Word Choice Model

We define a word choice model for word usage within a frame element. This additional factor is necessary to distinguish between various occurrences of the same type of frame element with different surface realizations. For example, assuming that the frame alignment model correctly aligns directionOfTurn attributes to a frame element of type DIRECTION, the word choice model will provide an additional weight for the alignment between the value of an attribute (e.g., "left") and the corresponding words within the frame element (e.g., "links"). Similarly to the word choice model within fields in (Liang et al., 2009), our model specifies a distribution over words given the attribute a. Depending on whether the attribute is typed for strings or categorial values, two different distributions are used.

String Attributes. For string attributes, we determine a weighting factor based on the longest common subsequence ratio (LCSR). The reason for using this measure is that we want to allow for spelling variants and the use of synonymous common nouns in the description of landmarks and street names (e.g., "Main St." vs. "Main Street", "Texas Steakhouse" vs. "Texas Restaurant"). The weighting factor $\mathrm{p}_{\mathrm{str}}(w|a)$ for an alignment pair $\langle f, a\rangle$ is a constant in the E-step and is calculated as the LCSR of the considered attribute value a_v and the content words $w = \mathrm{cw}(f)$ in an annotated frame element f divided by the sum over the LCSR values of all alignment candidates for a:

$$\mathrm{p}_{\mathrm{str}}(w|a) = \frac{\mathrm{LCSR}(a_v, w)}{\sum_x \mathrm{LCSR}(a_v, \mathrm{cw}(x))} \quad (5)$$

Categorial Attributes. We define categorial attributes as attributes that can only take a finite and prescribed set of values. For these we do not expect to find matching strings in NL directions as the attribute values are defined independently of the language in use (e.g., values for directionOfTurn are "left", "right" and "straight". However, the directions in our data set are in German, thus containing the lexemes "links", "rechts" und "geradeaus" instead). As the set of values $\{a_v \in \mathbb{D}_{a_t}\}$ for a categorial attribute type a_t is finite, we can define and train probability distributions over words for each of them during EM training. The models are initialized as uniform distributions and are used as a weighting factor in the E-Step. We re-calculate the parameters of a distribution $\mathrm{p}_{cat}(w|a)$ in each EM iteration by normalizing the estimated counts during M-step:

$$\mathrm{p}_{\mathrm{cat}}(w|a) = \frac{\mathrm{count}(a_v, w)}{\sum_x \mathrm{count}(a_v, x)} \quad (6)$$

5 Experiments and Results

5.1 Setting

We test the performance of different combinations of these EM-based models on our data, starting from a simple baseline model (**EM**), combined with the distance (**EM+dst**) and word choice models (**EM+wrd**) and finally the full model (**Full**). We perform additional experiments to examine the impact of different corpus sizes and an alignment threshold (**+thld**).

EM is a baseline model that consists of a simple EM implementation for aligning attributes and frame elements (equation (3)).

EM+dst consists of the simple EM model and the additional distance factor (equation (4)).

Model	P (+thld)	R (+thld)	F_1 (+thld)
Random	2.7 (2.7)	3.9 (3.9)	3.2 (3.2)
EM	2.0 (3.6)	2.9 (3.7)	2.34 (3.6)
EM+dst	7.3 (11.6)	10.8 (11.7)	8.7 (11.6)
EM+wrd	26.8 (36.3)	39.5 (35.5)	32.0 (35.9)
Full	28.9 (38.9)	42.5 (37.9)	34.4 (38.4)

Table 3: Precision (P), Recall (R) and F_1 measure results with and without threshold (+thld) on the alignment task (all numbers in percentages).

EM+wrd consists of the simple EM model with the word choice model (equations (5) and (6), respectively).

Full is the full alignment model including distance and word choice as described in Section 4 (cf. equation (1)).

We use the data set described in Section 3. The predictions made by the different models are evaluated against the gold standard alignments (cf. Tables 1 and 2). We run a total number of 30 iterations[4] of EM training on the complete data set to learn the parameters of the probability distributions. From the set of all possible 1-to-1 alignments, we select the most probable alignments according to the model in a way that no attribute and no frame element is aligned twice.

5.2 Results

We measure precision as the number of predicted alignments also annotated in the gold standard divided by the total number of alignments generated by our model. Recall is measured as the number of correctly predicted alignments divided by the total number of alignment annotations. As baselines we consider a random baseline (obtained from the average results measured over 1,000 random alignment runs) and the simple EM model.

The results in Table 3 show that the simple EM model performs below the random baseline. The individual extended models achieve significant improvement over the simple model and the random baseline. While the distance model has a smaller impact, the influence of the word choice model is considerable. Applying the full model yields further performance gains. We note that for all models recall is higher compared to precision.

[4]This number was determined by experiments as a general heuristics.

# directions	Precison	Recall	F_1
1	28.94%	42.31%	34.38%
2	29.04%	41.90%	34.31%
3	29.01%	42.18%	34.38%
4	28.75%	41.81%	34.07%
5	29.36%	42.69%	34.79%
6	30.18%	43.91%	35.77%

Table 4: Average results when using only a specific number of directions for each route with the model Full (-thld).

One of the reasons for this phenomenon may be that the EM-based models align as many attributes as possible to frame elements in the route directions. In our gold standard, however, only around 12% of all relevant attributes correspond to frame elements in the route directions (cf. Section 3.2). We estimate this quota from a part of the corpus and use it as an alignment threshold, i.e., for evaluation we select the best alignments proposed by the models, until we reach the threshold. With this we achieve a F_1 measure of 38.40% in a 6-fold cross validation test. This represents an improvement of 3.97 points and considerably boosts precision, yielding overall balanced precision (38.90%) and recall (37.92%).

A general problem of the current setup is the small amount of available data. With a total of 54 route directions, the data consists of 6 to 8 directions for each route. We compute a learning curve by using only exactly 1 to 6 directions per route to examine whether performance improves with increasing data size. The results are computed as an average over multiple runs with different data partitions (see Table 4). The results indicate small but consistent improvements with increasing data sizes, however, the differences are minimal. Thus we are not able to conclude at this point whether performance increases are possible with the addition of more data.

5.3 Error Analysis

In an error analysis on the results of the full model, we found that 363 out of 784 (46%) misalign-

ments are related to attributes not aligned in our gold standard. This is due to the fact that not all relevant attributes are realized in natural language directions. By addressing this problem in the model Full+threshold, we are able to reduce these errors, as evidenced by a gain of almost 10 points in precision and 4 points in F_1 measure.

We further observe that the word choice model does not correctly reflect the distribution of categorial attributes in the parallel corpus. In the data, we observe that humans often aggregate multiple occurrences of the same attribute value into one single utterance. An example of such a phenomenon can be seen with the attribute type 'directionOfTurn': Even though "straight" is the most common value for this attribute, it is only realized in directions in 33 (5%) cases (compared to 65% and 47% for "left" and "right" respectively). While our EM implementation maximizes the likelihood for all alignment probabilities based on expected counts, many pairs are not – or not frequently – found in the corpus. This results in the model often choosing incorrect alignments for categorial attributes and makes up for 23% of the misaligned attributes in total.

We found that further 5% of the attributes are misaligned with frame elements containing pronouns that actually refer to a different attribute. As our word choice model does not account for the use of anaphora, none of the affected frame elements are aligned correctly. Given the genre of our corpus, integrating simple heuristics to resolve anaphora (e.g., binding to the closest preceding mention) could solve this problem for the majority of the cases.

6 Conclusion

We presented a weakly supervised method for aligning route representations and natural language directions on the basis of parallel corpora using EM-based learning. Our models adopt ideas from Liang et al. (2009) with special adaptations to the current application scenario. As a major difference to their work, we make use of frame-semantic annotations on the NL side as a basis for segmentation.

While we can show that the extended models significantly outperform a simple EM-based model, the overall results are still moderate. We cannot draw a direct comparison to the results presented in Liang et al. (2009) due to the different scenarios and data sets. However, the corpus they used for the NFL recaps scenario is the closest to ours in terms of available data size and percentage of aligned records (in our case attributes). For this kind of corpus, they achieve an F_1 score of 39.9% with the model that is closest to ours (Model 2'). Their model achieves higher performance for scenarios with more available data and a higher percentage of alignments. Thus we expect that our model benefits from additional data sets, which we plan to gather in web-based settings.

Still, we do not expect to achieve near to perfect alignments due to speaker variation, a factor we also observe in the current data. As our ultimate goal is to generate NL instructions from given route representations, we can nevertheless make use of imperfectly aligned data for the compilation of high-confidence rules to compute semantic input structures for NLG. Following previous work by Barzilay and Lee (2002), we can also exploit the fact that our data consists of multiple directions for each route to identify alternative realization patterns for the same route segments. In addition, (semi-)supervised models could be used to assess the gain we may achieve in comparison to the minimally supervised setting.

However, we still see potential for improving our current models by integrating refinements based on the observations outlined above: Missing alignment targets on the linguistic side – especially due to anaphora, elliptical or aggregating constructions – constitute the main error source. We aim to capture these phenomena within the linguistic markup in order to provide hidden alignment targets. Also, our current model only considers frame elements as alignment targets. This can be extended to include their verbal predicates.

Acknowledgements: This work is supported by the DFG-financed innovation fund FRONTIER as part of the Excellence Initiative at Heidelberg University (ZUK 49/1). We thank Michael Bauer and Pascal Neis for the specification of the route representation format and Carina Silberer and Jonathan Geiger for annotation.

References

Anderson, Anne H., Miles Bader, Ellen Gurman Bard, Elizabeth Boyle, Gwyneth Doherty, Simon Garrod, Stephen Isard, Jacqueline Kowtko, Jan McAllister, Jim Miller, Catherine Sotillo, Henry Thompson, and Regina Weinert. 1991. The HCRC Map Task corpus. *Language and Speech*, 34(4):351–366.

Barzilay, Regina and Mirella Lapata. 2005. Collective content selection for concept-to-text-generation. In *Proceedings of the Human Language Technology Conference and the 2005 Conference on Empirical Methods in Natural Language Processing,* Vancouver, B.C., Canada, 6–8 October 2005, pages 331–338.

Barzilay, Regina and Lillian Lee. 2002. Bootstrapping lexical choice via multiple-sequence alignment. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing,* Philadelphia, Penn., 6–7 July 2002, pages 164–171.

Brown, Peter F., Vincent J. Della Pietra, Stephan A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19:263–311.

Burchardt, Aljoscha, Katrin Erk, Anette Frank, Andrea Kowalski, and Sebastian Pado. 2006. SALTO: A versatile multi-level annotation tool. In *Proceedings of the 5th International Conference on Language Resources and Evaluation,* Genoa, Italy, 22–28 May 2006, pages 517–520.

Dale, Robert, Sabine Geldof, and Jean-Philippe Prost. 2005. Using natural language generation in automatic route description. *Journal of Research and Practice in Information Technology*, 37(1):89–106.

Dempster, Arthur P., Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistics Society, Series B (Methodological)*, 39(1):1–38.

Duboue, Pablo A. and Kathleen R. McKeown. 2003. Statistical acquisition of content selection rules. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing,* Sapporo, Japan, 11–12 July 2003, pages 121–128.

Fillmore, Charles J., Christopher R. Johnson, and Miriam R.L. Petruck. 2003. Background to FrameNet. *International Journal of Lexicography*, 16(3):235–250.

Hansen, Stefan, Kai-Florian Richter, and Alexander Klippel. 2006. Landmarks in OpenLS: A data structure for cognitive ergonomic route directions. In *Proceedings of the 4th International Conference on Geographic Information Science,* Münster, Germany, 20-23 September 2006.

Liang, Percy, Michael Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of ACL-IJCNLP 2009*, pages 91–99, August.

Mani, Inderjeet. 2001. *Automatic Summarization*. John Benjamins, Amsterdam, Philadelphia.

Maxwell, John T. and Ronald M. Kaplan. 1993. The interface between phrasal and functional constraints. *Computational Linguistics*, 19(4):571–590.

Neis, Pascal and Alexander Zipf. 2008. Extending the OGC OpenLS route service to 3D for an interoperable realisation of 3D focus maps with landmarks. *Journal of Location Based Services*, 2(2):153–174.

Reiter, Ehud and Robert Dale. 2000. *Building Natural Language Generation Systems*. Cambridge, U.K.: Cambridge University Press.

Roth, Michael and Anette Frank. 2009. A NLG-based Application for Walking Directions. In *Companion Volume to the Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing,* Singapore, 2–7 August 2009, pages 37–40.

Schmid, Helmut. 1997. Probabilistic Part-of-Speech tagging using decision trees. In Jones, Daniel and Harold Somers, editors, *New Methods in Language Processing*, pages 154–164. London, U.K.: UCL Press.

Schuldes, Stephanie, Michael Roth, Anette Frank, and Michael Strube. 2009. Creating an annotated corpus for generating walking directions. In *Proceedings of the ACL-IJCNLP 2009 Workshop on Language Generation and Summarisation,* Singapore, 6 August 2009, pages 72–76.

Tiedemann, Jörg. 2003. Combining Clues for Word Alignment. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 339–346, Budapest, Hungary.

Vogel, Adam and Dan Jurafsky. 2010. Learning to Follow Navigational Directions. In *Proceedings of ACL-2010*, Uppsala, Sweden.

Vogel, Stephan, Hermann Ney, and Christoph Tillmann. 1996. HMM-based Word Alignment in Statistical Translation. In *Proceedings of the 16h International Conference on Computational Linguistics (COLING)*, pages 836–841, Copenhagen, Denmark.

A Multiple-Domain Ontology Builder

Sara Salem **Samir AbdelRahman**

Computer Science Department
Faculty of Computers and Information - Cairo University
{s.salem,s.abdelrahman@fci-cu.edu.eg}

Abstract

The interpretation of a multiple-domain text corpus as a single ontology leads to misconceptions. This is because some concepts may be syntactically equal; though, they are semantically lopsided in different domains. Also, the occurrences of a domain concept in a large multiple-domain corpus may not gauge correctly the concept significance. This paper tackles the mentioned problems and proposes a novel ontology builder to extract separate domain specific ontologies from such a corpus**.** The builder contribution is to sustain each domain specific concepts and relations to get precise answers for user questions. We extend a single ontology builder named Text2Onto to apply our thought. We fruitfully enhance it to answer, more precisely, questions on a subset of AQUAINT corpus.

1 Introduction

Domain ontology is a knowledge representation of the domain as a set of concepts and relations. Ontology notion always presents handy semantic solutions for various hot research areas such as Semantic Web, Informational Retrieval, and Question Answering.

Currently, automatic ontology builders presume that the given corpus has a single domain. When used with a multiple-domain corpus, these builders generate 1 large ontology for the whole corpus. Dramatically, this causes 2 domain misconception problems. First, the ontology conceptual model becomes imprecise for the common concepts in various domains having different semantics. Second, the relevance weights assigned to the concepts do not measure precisely their significance in specific domains.

This paper presents a promising solution for the mentioned problems. The proposed solution is an integrated 2-layer ontology builder. The ontology layers are: 1) the conceptual layer, which has the key concepts and relations of each separate domain, and 2) the general layer, which maintains the general domain information regarding related persons, organizations, locations, and dates. Our proposed 2-layer ontology improves the extracted answers for single-domain and cross-domain questions. We successfully prove our thought against Text2Onto builder.

Ontology extraction from a domain corpus has been targeted by many researchers. The core extraction approaches can be classified into 3 approaches. The first approach is to build the ontology from scratch (Buitelaar et al., 2004; Cimiano and Völker, 2005). The second approach is to extend a predefined general ontology, such as WordNet, with possible application domain concepts and relations (Navigli and Velardi, 2004). The last approach is to build ontology as a composition of other predefined ontologies (Cimiano et al., 2006). Moreover, as an ontology building design decision, the resultant ontology is either a single layer ontology or a multi-layered ontology (Benslimane et al., 2000; Dumontier and Villanueva-Rosales, 2007).

The paper is organized as follows: Section 2 introduces some related systems; Section 3 explains the misconceptions due to extracting a single ontology from a multiple-domain corpus; Section 4 describes our proposed builder; Section 5 illustrates our Question Answering system, which is used for the evaluation; Section 6 states our evaluation results; and Section 7 draws our conclusion and directions for the future work.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 967–975,
Beijing, August 2010

2 Related Work

There are 3 main approaches for ontology building, namely building from scratch, extending a general ontology, or building an ontology as a composition of other predefined ontologies.

Text2Onto (Cimiano and Völker, 2005) applies the first approach. It is a framework for learning ontologies automatically from textual data. It implements diverse linguistic and statistical techniques to extract domain concepts and relations. It combines results from different techniques, and it represents the extracted ontology elements in a so called Probabilistic Ontology Model (POM), which assigns a confidence value for each learnt element.

OntoLT (Buitelaar et al., 2004) is another example of building from scratch. It is a Protégé[1] plug-in that extracts ontology from text by defining a set of mapping rules. The rules map certain linguistic structures in an annotated text into ontological elements. The extracted elements are validated by the user before being inserted into the ontology.

OntoLearn (Navigli and Velardi, 2004) employs the second approach. It is a framework for trimming and extending general purpose ontologies, like WordNet, with specific domain terminologies and taxonomies. It extracts domain terminologies, and it uses a relevance measure to keep out non-relevant terms. OntoLearn uses a novel technique, called SSI, to assign a domain specific term to the correct sense in a general ontology.

The third approach is proposed in (Cimiano et al., 2006). It presents a system that integrates several heterogeneous semantic sources into 1 ontology, which is used to extract answers for user queries from various knowledge sources.

As a design decision, the ontology may consist of a single layer or of multiple layers. Benslimane et al. (2000) apply the multiple-layer approach for manually generating a set of interrelated ontology layers; each layer models a spatial domain specific function. Also, Dumontier and Villanueva-Rosales (2007) suggest a 3-layer ontology design. The first layer (primitive layer) defines the basic domain concepts and relations. The second layer (complex layer) imposes more complex domain restrictions on the primitive layer. The top layer (application layer) maintains application specific restrictions.

Our builder constructs a layered ontology from scratch. Its main distinguished features are: 1) generating separate domain specific ontologies from a multiple-domain corpus, 2) extracting general domain information, in addition to core domain conceptual information, and 3) it is an automatic multi-layered ontology builder, unlike other automatic builders, which generate single layer ontologies.

Our system can extend current builders, which extract ontologies from textual data, allowing them to handle a multiple-domain corpus. We selected Text2Onto because it is an automatic ontology builder, and it implements a variety of algorithms to extract many types of ontology elements. We use a news corpus as a multiple-domain corpus since it contains documents from different domains like Politics, Sports, Arts, and Finance.

3 Ontology Misconceptions

Building a single ontology for a given corpus is a familiar method. However, when dealing with a multiple-domain corpus, the builder usually suffers from the following 2 problems:

First, the ontology conceptual model becomes imprecise in the definition of common concepts that are semantically lopsided in different domains. For example, the concept "wall street" in the Finance domain is defined as a financial institution, and it is in the Arts domain defined as a movie. It is inaccurate to define the concept with 2 totally different meanings in 1 ontology. It is also incorrect to ignore a definition of them. When using Text2Onto for that example, it generates only 1 definition for "wall street" as a subclass-of "institution".

Second, when weighing concepts in a multiple-domain corpus, the relevance weights assigned to the concepts do not indicate the significance of each concept in a certain domain. As a result, some core domain specific concepts may have low weights with respect to the whole corpus. For example the concept "trading" has a low weight in a multiple-domain corpus; although, it is a main concept in the Finance domain (Section 6.2). This gives wrong indication of the concept importance to the user.

[1] http://protege.stanford.edu/

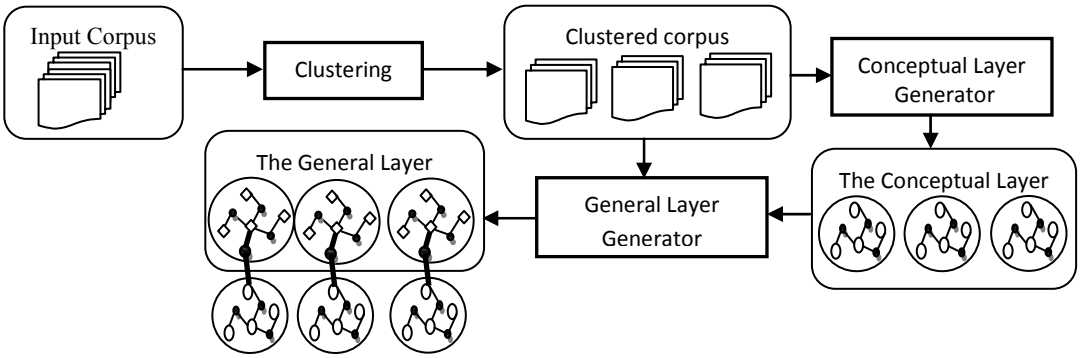


Figure 1. The Multiple-Domain Ontology Builder.

4 The Proposed Ontology Builder

Our builder aims to extract precise ontologies, which model possible knowledge in a multiple-domain corpus. A domain corpus, mostly, not only contains information about the core domain concepts and their relations, but it also contains general domain information such as dates of events and names of persons, locations, or organizations participating in the domain. Existing ontology builders either ignore this general information or they provide a limited implementation to extract it.

4.1 System Overview

The input to our builder (Figure 1) is a multiple-domain corpus. The first step is the ***clustering operation***, which divides the given corpus documents into clusters that are different among each other with high internal similarity. The next step is the ***conceptual layer generation***. In this step, we use Text2Onto to extract a separate ontology for each domain. Finally, the ***general layer generator*** uses each domain corpus and the conceptual layer ontology to extract relations among the concepts and the Named Entities in that domain.

4.2 The Conceptual Layer

The first step in constructing the conceptual layer is the clustering operation. We separate a multiple-domain corpus into various domain specific corpora such that the domain concepts are weighted based on their significance in that domain; also, the common concepts in different domains are separated. We favored a hierarchical clustering technique over a flat clustering one. That was because the number of resulting clusters should be known as a parameter in the latter. However, the number of corpus domains might be unknown in our case.

We employ the agglomerative hierarchical clustering technique (Manning et al., 2008). The technique starts with each document as a singleton cluster, and then it successively merges pairs of similar clusters until all clusters are merged into 1 cluster. We use the vector space model (Manning et al., 2008) to represent each document as a vector of terms' weights. The weight of a term *w* in a document *d* is calculated using the TF-IDF measure (Equation 1).

$$TFIDF(w,d) = TF(w,d) * log\frac{N}{DF(w)} \qquad (1)$$

Where *N* is the corpus size, *TF (w,d)* is the number of occurrences of the term *w* in the document *d*, and *DF (w)* is the number of documents containing the term *w*.

The similarity between 2 documents is calculated using the Cosine Similarity measure (Equation 2).

$$cosine(d1,d2) = \frac{V(d1)\,.\,V(d2)}{||V(d1)|| * ||V(d2)||} \qquad (2)$$

Where *V(d)* is the terms' weights vector for the document *d*, *||V(d)||* is the Euclidean length of the vector *V(d)*, and the numerator is the dot product of the 2 vectors.

The similarity between 2 clusters is calculated using the UPGMA measure (Steinbach et al., 2000) (Equation 3).

$$sim(C1, C2) = \frac{\sum_{d1 \in C1} \sum_{d2 \in C2} cosine(d1, d2)}{size(C1) * size(C2)} \quad (3)$$

We use the UPGMA measure to cluster a subset of DMOZ[2] data (1322 documents, in 4 domains), and it performs F-Measure of 0.86. Steinbach et al. (2000) describe how to calculate F-Measure for a hierarchy of clusters.

The combination similarity is the similarity of 2 merged clusters. We use this measure to cut the clusters hierarchy into M clusters by grouping ones having a minimum combination similarity of the threshold value €[3]. After clustering, we use Text2Onto to generate an ontology for each cluster (domain).

4.3 The General Layer

Text2Onto performs well in extracting ontology elements such as concepts, sub-class-of relations, instance-of relations, and part-of relations. Unfortunately, it performs inaccurately in extracting general domain information such as Named Entities and numeric information. There are 3 reasons for such misconception. First, proper nouns are not extracted as concepts. Second, numeric data is ignored. Third, restricted patterns are applied for the relations of Named Entities, that include only verb relations like [(NP |PNP) *verb* (NP|PNP)] and instance-of relations like [NP *such as* PNP], [*such* NP *as* PNP], and [PNP (NP)].

Because of the above reasons, we propose a highly flexible pattern based relation extractor. In our system, a pattern is a sequence of tags in the form of a regular expression. The possible tags are the normal POS tags like NN, VB, JJ, IN besides the following 5 tags CONCEPT, PERSON, LOCATION, ORGANIZATION, and DATE. This criterion is called Mixed Tagging. Currently, dates are the only data containing numbers extracted by our builder, but we can easily extend it to handle more numeric data.

The Mixed Tagging operation inputs are a document and the related conceptual ontology (Figure 2). The operation output is a mixed tagged document. The tagged text is then provided to the Relations Extractor to take out all

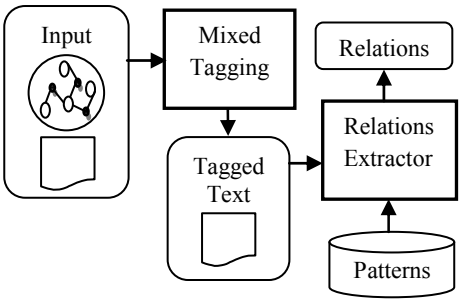


Figure 2. The General Relations Extraction.

relations matching our current predefined patterns. Example patterns are listed in Table 1; the first 2 patterns are verb relations, and the last 2 are noun relations.

The regular expression ([.{1,12}]){0,5} is used to limit the maximum number of tokens between the subject, the object, and the relation to 5 tokens. The expression [NN.?.?] matches any noun tag, and [VB.?] matches any verb tag.

After extracting the relations in all domain documents, the domain general ontology is created. It imports the corresponding conceptual ontology to model the relations among Named Entities and concepts.

([PERSON]) ([.{1,12}]){0,5}**([VB.?])+** ([.{1,12}]){0,5}**([CONCEPT])**
([ORGANIZATION])([.{1,12}]){0,5} **([DATE])**([.{1,12}]){0,5}**([VB.?])+**
([PERSON])([.{1,12}]){0,5} **([NN.?.?])+**([.{1,12}]){0,5}**([DATE])**
([NN.?.?])+([.{1,12}]){0,5}**([PERSON])** ([.{1,12}]){0,5}**([ORGANIZATION])**

Table 1. Sample Relation Patterns.

5 Question Answering System

Based on (Brank et al., 2005), a generated ontology can be evaluated using 4 different ways: 1) by a human who assesses it based on specific criteria, 2) by a comparison with the source data, 3) by a comparison with a golden standard, or 4) by using the ontology in an application and measuring the application performance. We chose the last option because the manual human assessment and the comparison with the source data are time consuming. Also, there is no golden standard ontology for a multiple-domain corpus.

Recently, researchers have studied the use of ontologies to extract answers to the user questions. AquaLog (Lopez et al., 2007) and

[2] http://www.dmoz.org/

[3] For clustering 600 AQUAINT documents, we use €=0.55 resulting in 7 Clusters (Secion 6.4).

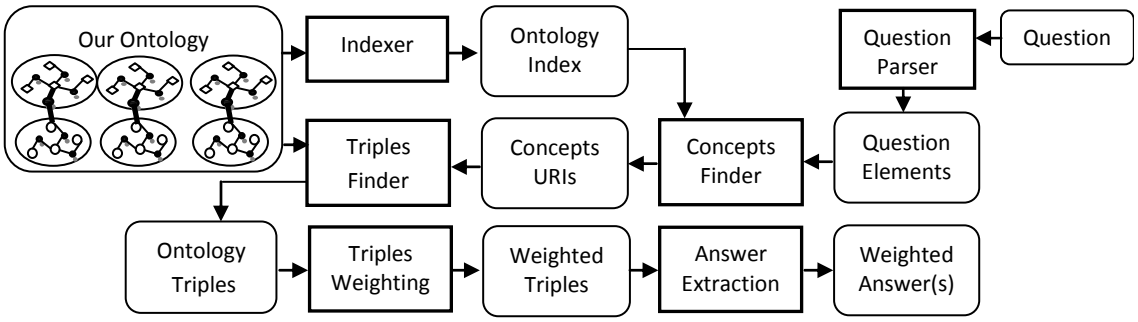


Figure 3. The Question Answering System.

PowerAqua (Lopez et al., 2009) are both ontology based Question Answering systems. PowerAqua extracts answers from various ontologies available on the web, unlike AquaLog, which extracts answers from 1 configurable ontology.

5.1 System Description

We implemented our simple Question Answering system handling who, when, where, and what questions. In the following, we describe the components of the system (Figure 3).

The Indexer: to make it easier for the system to locate the question concepts, an index is generated for our layered ontology. All concepts in different ontologies containing a certain stem are grouped in an index entry in the index file. The form of an index entry is as follows:

Stem,(Concept URI)+

The Question Parser: this component parses the user question, and it extracts 4 elements from it. First, the answer type; it can be PERSON, LOCATION, ORGANIZATION, DATE, or ANY based on the question type such as who, where, when, or what. Second, the answer restriction; it is used to limit the answers of *what* questions. For example, the answers for "what sport …?" question are restricted only to the sport types. Third, the question target; it defines the thing in which the question is interested. The fourth element is the relation; it contains the main verb(s) in the question. As an example, the elements of the question "*What sport does Jennifer Capriati play?*" are: the answer type (ANY), the restriction (sport), the question target (Jennifer Capriati), and the relation (play).

For a compound (2-clause) question such as *"What countries have Rhodes Scholars come from **and** has the Hale Bopp comet visible?*", each question clause is parsed as a separate question; finally, the answer extraction step intersects the answers of both clauses.

The Concepts Finder: using the ontology index, it locates concepts containing the stems of the question target and the restriction (if exists).

The Triples Finder: it extracts the triples which contain the question target concepts either as subjects or as objects. If the question is a definition question like "What is something?", the triple finder extracts only the sub-class-of triples.

The Triples Weighting: the triples are weighted based on their similarity to the question using our similarity criterion (Equation 4):

$$sim(Q,T) = \frac{\sum_{a \in Q} \underset{b \in T}{Lin(a,b)}}{L(Q) * L(T)} \quad (4)$$

Where Q and T are sets of the bag-of-words for the question relation and the triple relation respectively, $Lin(a,b)$ is a measure for the semantic similarity between a and b based on WordNet (Lin, 1998), and $L(x)$ is the number of elements in the set x.

The Answer Extraction: this component first filters out the triples mismatching the expected answer type. Then, if there is no restriction element, it extracts the answer from the weighted triples by considering the triple object if the question target is the subject, and vice versa. The extracted answer from a triple is assigned the same triple weight. If the question has a restriction element, the answer(s) will be limited to the sub concepts of the restriction element. A weight (Equation 5) is assigned to each sub concept s based on its similarity to the extracted triples as follows:

$$W(s) = \frac{\sum_{T \in R} sim(S,T)}{L(R)} \quad (5)$$

Where *R* is the set of extracted triples, *S* and *T* are the sets of bag-of-words for the sub concept and the triple relation respectively, *sim(S,T)* is calculated using Equation 4, and *L(R) is* the number of elements in *R*.

For a compound question, the list of resulting answers contains only the common answers extracted for the 2 clauses.

6 Evaluation and Discussion

In our evaluation, we assess: 1) the enhancement of the concepts' weights in a specific domain corpus, 2) the enhancement of modeling common concepts in different domains with different semantics, and 3) the performance of our Question Answering system. The assessment is done through a comparison between our approach and Text2Onto.

In the development of our builder, we used Text2Onto [4], Stanford Part-Of-Speech Tagger (POS Tagger)[5], Stanford Named Entity Recognizer (NER)[6], and Jena[7]. In the Question Answering system, we also used the Java WordNet Similarity Library (JWSL)[8]; it implements the Lin measure.

6.1 Data Set

Our evaluation is based on the AQUAINT[9] corpus (Graff, 2002). It is an English news corpus containing documents from the New York Times News Service, the Xinhua News Service, and the Associated Press Worldstream News Service. The Question Answering track in TREC[10] (The Text REtrieval Conference) provides a set of questions on AQUAINT corpus along with their answers.

6.2 Concepts Weights Enhancement

For this experiment, we generated a corpus for the 3 domains, namely Finance, Sports, and Movies, from AQUAINT documents, such that each domain has equal number of documents. We measured the concepts' significance weights when using Text2Onto to generate a single ontology for the whole corpus and when using our builder to generate 3 different domains ontologies. We consider 3 measures implemented in Text2Onto, namely the Relative Term Frequency (RTF), the Entropy, and the TF-IDF.

The RTF for a concept *w* is the probability of the concept occurrence in the corpus (Equation 6).

$$p(w) = \frac{No.\,of\ occurences\ of\ w}{No.\,of\ all\ corpus\ concepts} \quad (6)$$

The entropy and the normalized entropy for a concept *w* are calculated as follows (Equations 7 and 8 respectively):

$$Ent(w) = p(w) * \log p(w) \quad (7)$$

$$N\big(Ent(w)\big) = \frac{Ent(w)}{Min\ Ent - Max\ Ent} \quad (8)$$

In Section 4.2, we mention how to calculate the TF-IDF value for a term *w* in a document *d* (Equation 1). The TF-IDF weight and the normalized TF-IDF weight for a concept *w* in the whole corpus are calculated as follows (Equations 9 and 10 respectively):

$$TFIDF(w) = \frac{\sum_{d \in D} TFIDF(w,d)}{N} \quad (9)$$

$$N\big(TFIDF(w)\big) = \frac{TFIDF(w)}{\sqrt{\sum_{ci \in C} TFIDF(ci)^2}} \quad (10)$$

Where *D* is the set of documents containing *w*, *N* is the corpus size, and *C* is the set of all concepts in the corpus.

Since the concept weight is proportional to its occurrences in the corpus with respect to the other concepts, the fair distribution of the occurrences leads to precise weight calculation. In the specific domain corpus, the distribution is more reasonable than in multiple-domain corpus.

[4] http://code.google.com/p/text2onto/
[5] http://nlp.stanford.edu/software/tagger.shtml
[6] http://nlp.stanford.edu/software/CRF-NER.shtml
[7] http://jena.sourceforge.net/
[8] http://grid.deis.unical.it/similarity/
[9] http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2002T31
[10] http://trec.nist.gov/data/qa.html

Domain	Concept	Entropy		TF-IDF		RTF	
		Text2 Onto	Our Builder	Text2 Onto	Our Builder	Text2 Onto	Our Builder
Finance	Stock	0.181	0.999	0.053	0.103	0.001	0.020
	Trading	0.155	0.670	0.044	0.139	0.001	0.010
	Shares	0.100	0.670	0.036	0.139	0.000	0.010
	Economy	0.100	0.670	0.026	0.051	0.000	0.010
Sports	Sport	0.822	0.974	0.344	0.379	0.012	0.019
	Baseball	0.321	0.389	0.147	0.190	0.003	0.006
	League	0.299	0.363	0.134	0.174	0.003	0.005
	Football	0.205	0.251	0.085	0.111	0.002	0.003
Movies	Actor	0.525	0.613	0.150	0.194	0.007	0.022
	Movie Industry	0.230	0.362	0.098	0.263	0.002	0.011
	Music	0.205	0.326	0.085	0.230	0.002	0.009
	Home Video	0.038	0.066	0.012	0.032	0.000	0.001

Table 2. Concepts Weights Comparison between Our Builder and Text2Onto.

This fact can be verified easily from Table 2. The 3 measures give higher weights in the domain specific ontologies than in a single ontology for the whole corpus.

6.3 Modeling Common Concepts

To study the enhancement in modeling common concepts having different meaning in different domains, we chose 5 concepts as samples (Table 3). For each concept, we selected documents from AQUAINT and from the Wikipedia concerning the concepts in 2 different domains.

In this experiment, the single ontology generated by Text2Onto contains only 1 definition for each concept namely wall_street *is_a* institution, marijuana *is_a* drug, bear *is_a* mammal, jaguar *is_a* cat, and world_war *is_a* war. On the other hand, our builder maintains both concept definitions in different ontologies.

Concept	Definition 1	Definition 2
Wall Street	A financial Institution	A movie
Marijuana	A drug	A song
The bear	A Mammal	A movie
Jaguar	A big cat	A car
World War	A war	A museum

Table 3. Sample of Lopsided Concepts.

6.4 Question Answering Enhancement

The experiment includes common concepts definition questions, single-domain questions, and cross-domain questions.

To illustrate the effect of the common concepts misconception problem solved by our builder against Text2Onto, we generated 5 definition questions for the 5 concepts in Table 3, like "what is wall street?", "what is marijuana?"…etc.

For the single-domain questions, we used a subset of AQUAINT corpus composed of 600 documents clustered into 7 domains using combination similarity threshold value of 0.55. We selected 60 factoid questions from TREC 2004 questions having their answers in these documents. Examples of single-domain questions are:

- *Who discovered prions?*
- *When was the IFC established?*

In addition to factoid questions, TREC 2004 also includes list questions. The answers of each question are aggregated from multiple documents. We used these questions in generating 10 cross-domain questions. Each question combines 2 of TREC list questions such that the 2 list questions are in different domains. Examples of these questions are:

- *What cities have an Amtrak terminal and have Crip gangs?*
- *What countries are Burger King located in and have IFC financed projects?*

Evaluation Criteria: the accuracy (A) (Equation 11) is used for evaluating single-domain questions because each factoid question has only 1 correct answer.

$$A = \frac{No.of\ correct\ answers}{No.of\ questions} \quad (11)$$

The definition and cross-domain questions have multiple correct answers. The average Precision (P), Recall (R), and F-Measure (F) (Equations 12, 13, and 14 respectively) of all questions are used for our evaluation.

$$P = \frac{No.of\ correct\ answers}{No.of\ retrieved\ answers} \quad (12)$$

$$R = \frac{No.of\ correct\ answers}{No.of\ actual\ answers} \quad (13)$$

$$F = \frac{2 * P * R}{P + R} \quad (14)$$

Table 4 shows that, in the definition questions, we achieve F-Measure of 1, while Text2Onto achieves 0.5. This is because our builder maintains the 2 different definitions of each concept, unlike Text2Onto, which contains only one.

Questions Type	Our Ontology	Text2Onto Ontology
Definition Questions	P=1.0 R=1.0 F=1.0	P=0.5 R=0.5 F=0.5
Single-Domain	A=68%	A=0.05%
Cross-Domain	P=0.49 R=0.59 F=0.44	P=0 R=0 F=0

Table 4. Question Answering Evaluation.

In the single-domain questions, using our ontology, we could answer 41 questions while using Text2Onto ontology we could answer only 3 questions ("what particle is a quark?", "what are prions made of?", and "What is the treatment of cataract?"). The low coverage of Named Entities in Text2Onto hinders it from answering correctly any question of types Who, When, and Where. This indicates the enhancement introduced by the proposed general layer for modeling accurately more domain information. In the cross-domain questions, we achieve F-Measure of 0.44. None of the cross-domain questions are answered using Text2Onto ontology due to the mentioned Named Entity coverage problem.

Although our results are better than Text2Onto, there is a room for more improvements. There are 4 main sources for retrieving wrong or incomplete answers (Table 5). Some relations are not extracted because their elements (subject, relation, and object) are not near enough from each other in the text, so none of our patterns or Text2Onto patterns could match them. This is the source of 65% of the errors. Missed Named Entities or wrongly tagged ones cause 16% of the errors. Some relations are not extracted because co-reference has not been handled yet. That leads to 12% of the total errors. Finally, in the factoid questions, we consider the answer with the highest weight to be the correct answer; 7% of the answers are extracted but with lower weights.

Error Type	Error percentage
No matching pattern	65%
NER Error	16%
Co-Reference	12%
Low answer weight	7%

Table 5. Answer Error Sources.

Based on the mentioned experiments, our builder outperforms Text2Onto in Question Answering. In addition, it can be used skillfully to enhance other Natural Language Processing applications such as Information Retrieval from multiple-domain data. Our initial results using 220 queries on 600 AQUAINT documents records 0.35 F-Measure against Lucene[11], which achieves 0.18.

7 Conclusion and Future Work

This paper presents the misconception problems when interpreting a multiple-domain corpus in a single ontology. A novel ontology builder is presented handling these problems by generating separate domain ontologies describing core and general domain information.

Currently, we hand on improving our builder relation extractor to answer more TREC questions by automatically learning patterns from text and by handling co-reference. Moreover, we are working to enhance the performance of our Information Retrieval system.

[11] http://lucene.apache.org/

References

Benslimane, D., E. Leclercq, M. Savonnet, M.-N. Terrasse, and K. Yétongnon. 2000. *On the Definition of Generic Multi-layered Ontologies for Urban Applications.* In the International Journal of Computers, Environment, and Urban Systems, volume 24: 191-214.

Brank, Janez, Marko Grobelnik, and Dunja Mladenić. 2005. *A Survey of Ontology Evaluation Techniques.* In the Proceedings of the 8th International Multiconference on Information Society: 166-169.

Buitelaar, Paul, Daniel Olejnik, and Michael Sintek. 2004. *A Protégé Plug-In for Ontology Extraction from Text Based on Linguistic Analysis.* In the Proceedings of the 1st European Semantic Web Symposium: 31-44.

Cimiano, Philipp, and Johanna Völker. 2005. *Text2Onto - A Framework for Ontology Learning and Data-driven Change Discovery*. In the Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems: 227-238.

Cimiano, Philipp, Peter Haase, York Sure, Johanna Völker, and Yimin Wang. 2006. *Question Answering on Top of the BT Digital Library.* In the Proceedings of the 15th International Conference on World Wide Web: 861-862.

Dumontier, Michel, and Natalia Villanueva-Rosales. 2007. *Three-Layer OWL Ontology Design.* In the Proceedings of the *2nd* International Workshop on Modular Ontologies. CEUR Workshop Proceedings, volume 315.

Graff, David. 2002. *The AQUAINT Corpus of English News Text*. Linguistic Data Consortium, Philadelphia.

Lin, Dekang. 1998. *An Information-Theoretic Definition of Similarity*. In the Proceedings of the 15th International Conference on Machine Learning: 296-304.

Lopez, Vanessa, Victoria Uren, Enrico Motta, and Michele Pasin. 2007. *AquaLog: An Ontology-driven Question Answering System for Organizational Semantic Intranets*. In the Journal of Web Semantics, volume 5: 72-105.

Lopez, Vanessa, Victoria Uren, Marta Sabou, and Enrico Motta. 2009. *Cross Ontology Query Answering on the Semantic Web: An Initial Evaluation*. In the Proceedings of the 5th International Conference on Knowledge Capture: 17-24.

Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Online edition. http://nlp.stanford.edu/IR-book/information-retrieval-book.html. Cambridge University Press.

Navigli, Roberto, and Paola Velardi. 2004. *Learning Domain Ontologies from Document Warehouses and Dedicated Web Sites.* In the Journal of Computational Linguistics, volume 30: 151-179.

Steinbach, Michael, George Karypis, and Vipin Kumar. 2000. *A Comparison of Document Clustering Techniques.* Technical Report #00-034, University of Minnesota.

Weakly Supervised Morphology Learning for Agglutinating Languages Using Small Training Sets

Ksenia Shalonova
Computer Science,
University of Bristol
ksenia@cs.bris.ac.uk

Bruno Golénia
Computer Science,
University of Bristol
csbsgg@bristol.ac.uk

Abstract

The paper describes a weakly supervised approach for decomposing words into all morphemes: stems, prefixes and suffixes, using wordforms with marked stems as training data. As we concentrate on under-resourced languages, the amount of training data is limited and we need some amount of supervision in the form of a small number of wordforms with marked stems. In the first stage we introduce a new Supervised Stem Extraction algorithm (SSE). Once stems have been extracted, an improved unsupervised segmentation algorithm GBUMS (Graph-Based Unsupervised Morpheme Segmentation) is used to segment suffix or prefix sequences into individual suffixes and prefixes. The approach, experimentally validated on Turkish and isiZulu languages, gives high performance on test data and is comparable to a fully supervised method.

1 Introduction

The major function of morphological analysis is decomposition of words into their constituents - stems and prefixes/suffixes. In recent years Machine Learning approaches were used for word decomposition. There is a number of both unsupervised morphology learning systems that use "raw" wordforms as training data (Creutz and Lagus, 2002; Goldsmith, 2001; Kazakov and Manandhar, 2001) and supervised morphology learning systems using segmented wordforms into stems and affixes as training data (Oflazer et al., 2001). The supervised morphology learning systems are usually based on two-level morphology (Koskenniemi, 1983). There is also a weakly supervised approach that uses, for example, wordpairs as input, and this was applied mainly to fusional languages for stem extraction (Erjavec and Dzeroski, 2004). Our project concerns developing speech technology for under-resourced languages. For this type of applications we need a relatively fast, cheap (i.e. does not require large training sets), almost knowledge-free approach that gives high performance. We have chosen to use wordforms with marked stems as training data in order to fulfill the criteria mentioned above.

Morphological analysis is used in many practical Natural Language Processing applications such as Machine Translation, Text Mining, spell-checkers etc. Our near-term goal is the integration of the morphology learning algorithms into the language-independent Text-to-Speech (TTS) system for improvement of grapheme-to-phoneme rules, stress prediction and tone assignment. In particular, the morphology learning algorithms described in this paper will be incorporated into the available isiZulu TTS system for automatic prediction of lexical tones. In the isiZulu language lexical tone assignment depends on the morpheme boundary. The current isiZulu TTS system is tone-deaf due to the lack of morphological decomposition. A number of perception tests will be carried out in order to evaluate which performance of morphology decomposition is acceptable for TTS and will improve the quality of the synthesised speech. It seems that the unsupervised morphology learning systems can be relatively easy to implement from scratch, but their performance probably cannot be regarded as high enough to improve the performance of the synthesized speech. In order to overcome this problem we present a novel synthesis of supervised and unsupervised induction techniques for morphology learning.

Our approach consists of two parts: the new supervised stem extraction algorithm for agglutinat-

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 976–983,
Beijing, August 2010

ing languages and the improved version of the unsupervised algorithm for segmenting affix sequences. In (Shalonova et al., 2009) the authors presented the function learning approach called TASR (Tree of Aligned Suffix Rules) for extracting stems in fusional languages given wordpairs (word in grammatical form - word in basic form). While this algorithm gives good performance for Russian and English, it gives quite poor performance for agglutinating languages as shown in Section 4. A new approach for stem extraction in agglutinating languages is required for two main reasons. Firstly, suffix (or prefix) sequences in agglutinating languages can be much longer than in fusional languages and TASR does not seem to be efficient on long affix sequences as it does not generalise data in the efficient way and generates too many specific rules. This leads to poor performance on unseen data. Secondly, in some agglutinating languages it could be easier for native speakers to provide a stem (i.e. to provide a list of wordforms with annotated stems), whereas in highly inflective fusional languages the stem is often strongly bound with suffix sequences, and providing a proper stem requires high linguistic expertise. TASR approach is more appropriate for word-and-paradigm or realizational morphology that focuses on the whole word form rather than on word segmentation. For example, in Russian the infinitive verb *govorit'* ('to speak') generates a set of grammatical forms or a paradigm - *govorivshij*, *govor'aschij*, *govorim* etc.

The second part of our approach is the improved version of GBUAS algorithm (Shalonova et al., 2009) that provides affix segmentation given unannotated affix sequences. Given stem boundaries in the training set, our method splits the input word into all morphemes: stems and prefixes/suffixes. Our two-stage approach is tested on the under-resourced language isiZulu containing both prefixes and suffixes, as well as on Turkish containing only suffixes. Turkish is the most commonly spoken of the Turkic languages (over 77 million people). isiZulu is the Bantu language with about 10 million speakers and it is the most widely spoken home language in South Africa. Both Turkish and isiZulu use agglutination to form new words from noun and verb stems. In comparison to Turkish, isiZulu is a tonal language. In contrast to East Asian languages, in isiZulu there are three steps for tone assignment: lexical, morphemic and terraced. For TTS the lexical and morphemic tones will need to be recovered from the lexicon and the grammar as the orthography has no tone marking. The terraced tone relation can in general be recovered and marked automatically from the tone sequence with a finite state model.

2 Stem Extraction Algorithm

The Stem Extraction Algorithm (SSE) is the supervised algorithm for stem extraction. The training data for the SSE represent wordforms with the marked stem boundaries. During the training stage we collect a set of all possible stem extraction rules from training data and assign precision measures to each rule. Each rule is of the form L_R where "_" is the stem boundary, L and R are the left and right graphemic contexts of a stem boundary of different lengths. We differentiate prefix $Lpref\_Rstem$ and suffix $Lstem\_Rsuff$ stem extraction rules that correspond to the rules containing the left-hand stem boundary and the right-hand stem boundary respectively. For example, the Turkish word *yer* ('earth') with the marked word boundary #ye_r# generates the following $Lstem\_Rsuff$ rules: #ye_r#, #ye_r, ye_r#, #ye_, ye_r, e_r#, _r#, ye_, e_r, _r, and e_, where the symbol '#' signifies the word initial and final positions. We are implementing similar feature vectors used for automatic pronunciation prediction based on the focal grapheme (in our case it is a stem boundary) and left/right graphemic contexts of different length (Davel and Barnard, 2008). The idea of implementing expanding context in NLP tasks is usually applied for two-level data like grapheme-to-phoneme mapping rules (Torkkola, 1993), whereas in our case we use it for one-level data.

The precision measure for each rule is calculated by the formula $p/(p+n+\varepsilon)$ where p and n are the number of positive and negative examples, and ε is used to cover the cases where there are no negative examples. A high precision is desirable and this occurs when there are high values of p and low values of n (i.e. many positive examples and

few negative examples). Using negative examples in contrast to using only rule frequencies (or positive examples) improves the performance of the algorithm.

Definition 1. *The number of positive examples for the rule Lstem_Rsuff (or rule Lpref_Rstem) is the number of training instances of Stem_Suffixes (or Prefixes_Stem) containing the substring L_R.*

Definition 2. *The number of negative examples for rule Lstem_Rsuff (or Lpref_Rstem) is the number of training instances Stem_Suffixes (or Prefixes_Stem) such that Stem + Suffixes (or Prefixes + Stem) contains substring L+R and Stem_Suffixes (or Prefixes_Stem) does not contain substring L_R where '+' denotes string concatenation.*

In the above definitions '_' is a stem boundary.

Example 1. *Suppose we have only three isiZulu verbs: zi_bek_e, zi_nak_eke and a_hlul_eke. For the Lstem_Rsuff rule **'ek_e'**, the word zi_bek_e generates one positive example and the two other words zi_nak_eke and a_hlul_eke generate one negative example each.*

The approach given in Algorithm 1 aims to find the unique longest rule-pair '*Lpref_Rstem* and *Lstem_Rsuff*' with the highest precision that is applied to the input wordform for stem extraction. In case the language does not have prefixes like Turkish, the longest rule *Lstem_Rsuff* with the highest precision is applied. The decision of using either a rule-pair or just a single suffix rule is influenced by prior knowledge that a particular language has got either both prefixes and suffixes like isiZulu or only suffixes like Turkish. From now on we will use the term 'rulepair' in application both to the rulepair '*Lpref_Rstem* and *Lstem_Rsuff*' in case of isiZulu and to the rulepair ' and *Lstem_Rsuff*' with an empty first element in case of Turkish.

Algorithm 1 Choosing rule pair for stem extraction.

input W = raw wordform; P and S are sets of unique *Lpref_Rstem* and *Lstem_Rsuff* rules
output *result_rule_pair*

```
result_rule_pair ← ∅
iMaxlength ← ∞
repeat
  (p1,s1) ← getrulepair (P × S, W, iMaxlength)
  (p2,s2) ← getrulepair (P × S \ (p1,s1), W, iMaxlength)
  iMaxlength ← length(p1,s1)
until (p1,s1) = ∅ or precision (p1,s1) ≠ precision(p2,s2) or length (p1,s1) ≠ length(p2,s2)
result_rule_pair ← (p1,s1)

function getrulepair(PS, W, iMaxlength)
ilength ← 0
r ← ∅
for all (p,s) ∈ PS do
  if (p,s) matches W then
    if length(p,s) < iMaxlength and length(p,s) > ilength then
      ilength ← length(p,s)
      r ← (p,s)
    else
      if length(p,s) = ilength and precision(p,s) > precision(r) then
        r ← (p,s)
      end if
    end if
  end if
end for
return r
end function
```

The search is carried out on the set of rule pairs matching an input raw wordform. The set is sorted by length first, and then by precision measure within each length category.

For example, if rulepairs have the following length-precision values:
'4-0.5,'4-0.5','4-0.2'

'3-0.4','3-0.3'
'2-0.3'
rulepair with the value 3-0.4 is selected.

The rulepair matches the input word if *Lpref_Rstem* and *Lstem_Rsuff* rules can be applied without contradicting each other. For example, the rule pair '#a_hl' and 'l_eke' matches the word *a_hlul_eke*, whereas the rule pair '#a_hlulek' and 'le_ke' does not match this word. For each input wordform the set of its own rulepair candidates is generated. The search in the algorithm among these rulepairs starts from the longest rulepairs, and this allows more specific rules and exceptions to be applied first, whereas the more general rules are applied if no specific rules cover the input wordform.

3 Graph-Based Unsupervised Morpheme Segmentation

In this section we extend GBUMS (Graph-Based Unsupervised Morpheme Segmentation) that segments sequences of prefixes and suffixes (Golénia et al., 2009). We propose an extension of GBUMS which uses the graph structure of GBUMS through a brute-force method. Our experiments showed the improved results on training set and allowed GBUMS to be run on the test sets for two languages: Turkish and isiZulu.

The algorithm GBUMS was originally developed in (Shalonova et al., 2009) under the name GBUSS (Graph-Based Unsupervised Suffix Segmentation) to extract suffix sequences efficiently and it was applied to Russian and Turkish languages on training sets. We refer to prefixes and suffixes generally as morphemes. GBUMS uses a morpheme graph in a bottom-up way. Similar to Harris (Harris, 1955), the algorithm is based on letter frequencies. However, when Harris uses successor and predecessor frequencies, they use position-independent n-gram statistics to merge single letters into morphemes until a stopping criterion is fulfilled.

In the morpheme graph, each node represents a morpheme and each directed edge the concatenation of two morphemes labelled with the frequencies in a M-corpus (see Figure 1). M-corpus is a list of morpheme sequences

Definition 3. *Let $M = \{m_i | 1 \leq i \leq |M|\}$ be a set of morphemes, let f_i be the frequency with which morpheme m_i occurs in a M-corpus of morpheme sequences, let $v_i = (m_i, f_i)$ for $1 \leq i \leq n$, and let $f_{i,j}$ denote the number of morpheme sequences in the corpus in which morpheme m_i is followed by morpheme m_j. The* morpheme graph *$G = (V, E)$ is a directed graph with vertices or nodes $V = \{v_i | 1 \leq i \leq |V|\}$ and edges $E = \{(v_i, v_j) | f_{i,j} > 0\}$. We treat $f_{i,j}$ as the label of the edge from v_i to v_j.*

In G, each node is initialised with a letter according to a M-corpus, then one by one, nodes are merged to create the real morphemes. To merge nodes, an evaluation function is required. In (Golénia et al., 2009), Golenia et al. employed the *Morph_Lift* evaluation function based on its relation to the *lift* of a rule for association rules in data mining (Brin et al., 1997).

Definition 4. *$Morph\_Lift$ is defined for a pair of morphemes m_1 and m_2 as follows:*

$$Morph\_Lift(m_1, m_2) = \frac{f_{1,2}}{f_1 + f_2} \qquad (1)$$

From now on, we know how to merge nodes. Now, we need to figure out the most important part of GBUMS, which is the stopping criterion. The stopping criterion is to prevent overgeneralisation. In other words, the algorithm needs to be stopped before getting the initial M-corpus (since no merging is possible). This criterion is based on the Bayesian Information Criterion (BIC) and Jensen-Shannon divergence (Li, 2001).

BIC is used for selecting a model (set of morphemes) which fits a data set (M-Corpus) without being too complex. We want to point out that BIC is related to MDL. BIC is a trade-off between the maximum likelihood, the parameters of the model (probability and length of each morpheme) and the number of elements in the data set (frequency of each morpheme). A smaller value of BIC corresponds to a better model fit. The maximum of the Jensen-Shannon divergence is used in order to analyse the increase of log-likelihood among all possible models. The Jensen-Shannon divergence is defined as follows (Dagan et al., 1997):

Definition 5. *The* Jensen-Shannon divergence *is defined for two morphemes m_1 and m_2 as the de-*

crease in entropy between the concatenated and the individual morphemes:

$$D_{JS}(m_1,m_2) = H(m_1 \cdot m_2) - \frac{L_{m_1}H(m_1) + L_{m_2}H(m_2)}{N} \quad (2)$$

where $H(m) = -P(m)\log_2 P(m)$ $N = \sum_m Freq(m)$ *and* L_m *is the string length of m.*

Stopping criterion *requires that* $\Delta BIC < 0$ *which translates to:*

$$\max_{m_1,m_2} D_{JS}(m_1,m_2) \leq 2\log_2 N \quad (3)$$

Algorithm 2 The GBUMS morpheme segmentation algorithm

```
input  M-Corpus = List of Strings
output  M-CorpusSeg = List of Strings
  M-CorpusSeg ← SegmentInLetters(M-Corpus);
  Graph ← InitialiseGraph(M-CorpusSeg);
  repeat
    Max ← 0;
    for all (p,q) ∈ Graph do
      ML_Max ← Morph_Lift(p, q);
      if ML_Max > Max then
        Max ← ML_Max;
        pMax ← p;
        qMax ← q;
      end if
    end for
    Graph ← MergeNodes(Graph, pMax, qMax);
    M-CorpusSeg ← DeleteBoundaries(M-CorpusSeg, Label(pMax), Label(qMax));
    Graph ← AdjustGraph(M-corpusSeg, Graph);
  until StoppingCriterion(pMax, qMax, Max)
```

After several merging iterations, the output of the algorithm is the graph shown in Figure 1. The GBUMS is presented in Algorithm 2.
Note that the M-Corpus is completely segmented at the beginning of the algorithm. Then, the boundaries in the segmented M-Corpus are removed step by step according to a pair found in the graph with the maximum value for $Morph\_Lift$. When the stopping criterion is fulfilled, the segmented M-Corpus represents the morpheme sequences.

At this point we present our extension of GBUMS based on a brute-force heuristic which scores every possible segmentation of an input morpheme sequence using graph values. We consider the morpheme graph as a model where each morpheme sequence can be extracted by the *MGraph* function (eq. 4).

Definition 6. *We define MGraph of a morpheme sequence without boundaries x as follows:*

$$MGraph(x) = \arg\max_{t \subseteq x} \frac{1}{N_t - C_t} \sum_{m \in t} L_m log(f_m + 1) \quad (4)$$

where

- *t is a morpheme sequence with boundaries of x,*
- *m is a morpheme of t,*
- f_m *is the frequency of the morpheme m,*
- N_t *is the number of morphemes existing in the graph,*
- C_t *is the number of morphemes existing and contiguous in the graph.*

Firstly, as a post-processing procedure the *MGraph* function improves the performance on training data. Secondly, it permits the identification of unseen morphemes. That is why the model generated by GBUMS can be run on test data sets.

Example 2. *Let our final morpheme graph be as shown in Figure 1 where nodes represent suffixes and their frequencies.*
Let x="ekwe" be our input suffix sequence that we want to segment into individual suffixes. We split this input sequence into all possible substrings from individual characters up to size of the input string length: e-k-w-e, e-k-we, e-kw-e, ek-w-e, . . . , ekwe.
Using equation 4, we evaluate each substring and select the one with the highest score as the correct segmentation. Here, we have 7 potential segmentations with a score higher than 0 (MGraph > 0), e.g: e-k-w-e = $(\log(3)+\log(3))/2 = 1.0986$, *ek-w-e* =$(2\log(4)+\log(3))/2 = 1.9356$ *and ek-we* =

$2\log(4) = 2.7726$.
Consequently, ek-we is chosen as the correct segmentation for the substring "ekwe".

We would like to highlight that our new method can identify unseen cases with M-Graph, for instance, in the previous example suffix "we" was not present in the training graph, but was correctly extracted.

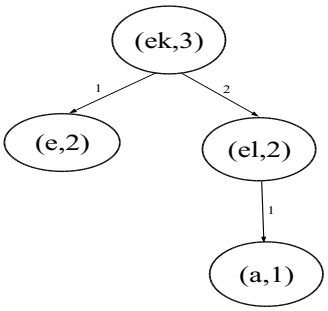


Figure 1: Example of a suffix subgraph in the training phase for isiZulu.

4 Results

Our experiments were based on Turkish data containing 1457 verbs and 2267 nouns, and isiZulu data containing 846 nouns and 931 verbs, with one single unambiguous segmentation per word.[1] Both isiZulu and Turkish data were uniquely sampled from the most frequent word lists.

Our first experiments compared TASR and the new SSE algorithm for stem extraction (10-fold cross validation assumes the following training and test set sizes: training sets containing 1311 wordforms for verbs and 2040 wordforms for nouns; test sets containing 146 wordforms for verbs and 227 wordforms for nouns). As can be seen from the Table 1, the performance of the SSE algorithm on Turkish data is much higher than that of TASR on the same dataset. As we mentioned in Section 1, TASR is not suitable for agglutinating languages with long suffix sequences. Although TASR algorithm gives an excellent performance on Russian, for most Turkish words it fails to extract proper stems.

[1] In agglutinating languages some wordforms even within one POS category can have several possible segmentations.

	Test	**FMea**
TASR	Nouns	20.7±6.8
	Verbs	12.6±5.9
SSE	Nouns	84.3±3.2
	Verbs	82.1±3.7

Table 1: Comparison of TASR and SSE for Turkish using 10-fold cross validation.

Our next experiments evaluated the performance of GBUMS on its own given unsegmented suffix sequences from Turkish nouns and verbs as training data. The performance on these training data increased by approximately 3-4 % in comparison to the results presented in (Shalonova et al., 2009). We would like to point out that the results in (Shalonova et al., 2009) are based on training data rather than on test data, whereas in the current paper we run our algorithms on test (or unseen) data. Our final experiments examined performance on the test sets and were run both on Turkish and isiZulu data. We compared our approach with Morfessor run both in supervised and in unsupervised mode. Although Morfessor is known as one of the best unsupervised morphology learning systems, it is possible to run it in the supervised mode as well (Spiegler et al., 2008). The training data for SSE+ GBUMS contained wordforms with marked stems. During training stage the SSE algorithm was collecting information about stem boundaries and the GBUMS algorithm was run on unlabelled suffix and prefix sequences from the same training set. The test stage for the SSE+GBUMS approach was run on "raw" wordforms by applying the SSE algorithm first for stem extraction and then running GBUMS algorithm for segmenting prefix or suffix sequences after the SSE has extracted stems. Training data for supervised Morfessor used the same wordforms as for the SSE+GBUMS training set and contained wordforms segmented into stems and affixes (i.e. words segmentated into all morphemes were given as training data). The test data for supervised Morfesor were the same as those used for SSE+GBUMS. Morfessor in unsupervised mode was run on "raw" wordforms as training data. To evaluate our current work we ap-

	Test	FMea
Supervised Morfessor	Nouns	74.6±2.3
	Verbs	84.5±2.2
SSE+ GBUMS	Nouns	78.8±2.4
	Verbs	76.9±0.7
Unsupervised Morfessor	Nouns	26.6±2.6
	Verbs	28.4±2.8

Table 2: Comparison of Morfessor and SSE+GBUMS for Turkish using 10-fold cross validation.

	Test	FMea
Supervised Morfessor	Nouns	76.7±1.6
	Verbs	88.5±2.4
SSE+ GBUMS	Nouns	87.9±1.9
	Verbs	84.5±2.5
Unsupervised Morfessor	Nouns	27.4±5.1
	Verbs	26.9±5.0

Table 3: Comparison of Morfessor and SSE+GBUMS for isiZulu using 10-fold cross validation.

plied the SSE+GBUMS approach for the under-resourced agglutinating language isiZulu containing both prefixes and suffixes and for Turkish containing only suffixes. The results show (Table 2 and Table 3) that our weakly supervised approach is comparable with the supervised Morfessor and decisively outperforms the unsupervised Morfessor. We think that it is useful to point out that unsupervised morphology learning systems in general require much larger training sets for better performance. F-measure is the harmonic mean of precision and recall, whereas precision is the proportion of true morpheme boundaries among the boundaries found, recall is the proportion of boundaries found among the true boundaries. In our experiments the GBUMS algorithm had no restrictions on affix length (Shalonova et al., 2009), but if there were restrictions, performance could be better. For isiZulu nouns our approach significantly outperformed supervised Morfessor, whereas for Turkish verbs SSE+GBUMS performed much worse. The best overall results obtained by GBUMS were based on the isiZulu nouns where about **53%** of all affixes were single letter affixes, whereas the worst results our approach gave for Turkish verbs where only about **12%** of affixes are composed of one letter. It is important to notice that the GBUMS algorithm, which is completely unsupervised, gives better results for extracting one letter affixes compared to Morfessor.

5 Conclusions

In the paper we described a weakly supervised approach for learning morphology in agglutinating languages. We were successful in our ultimate goal of synthesis of supervised and unsupervised induction techniques by achieving high performance on small amount of training data. Our weakly supervised approach is comparable with the supervised morphology learning system. As we are working with the languages for which linguistic resources are very limited (in particular words with morpheme boundaries), the developed method fulfills our goals of providing key components for speech and language products for such under-resourced languages. We speculate that the current performance might be improved by adding a small amount of completely ”raw” data to the training set.

The integration of our algorithms into working TTS systems is of key importance. As our near-term goal is the integration of morphology learning component into the currently working isiZulu TTS system, we will have to analyse the necessity of a Part of Speech Tagger (POS) and morphological disambiguation. In agglutinating languages some wordforms can be segmented in different ways (i.e. have different surface forms) and Machine Learning approaches normally select the most probable segmentation, and therefore our morphology disambiguation can be important. Morphological disambiguation for TTS can be considered a less complex problem than full morphological disambiguation as it can be linked, for example, to lexical tone disambiguation that may not require the full POS tag set. We intend to carry out user perception tests in order to evaluate the possible improvement in the isiZulu TTS quality after morphology information is added.

6 Acknowledgment

We would like to thank Kemal Oflazer from Sabanci University in Istanbul for his help and the Turkish data. We also thank Viktor Zimu and Etienne Barnard from CSIR in Pretoria for providing us isiZulu data. We also thank Roger Tucker for his support in the project. The work was sponsored by the EPSRC Grant EP/E010857/1 'Learning the morphology of complex synthetic languages'.

References

Brin, S., R. Motwani, J. Ullman, and S. Tsur. 1997. Dynamic itemset counting and implication rules for market basket data. In *ACM SIGMOD international conference on Management of data*, pages 255–264. ACM.

Creutz, M. and K. Lagus. 2002. Unsupervised discovery of morphemes. *Proceedings of the Workshop on Morphological and Phonological Learning of ACL-02*, pages 21–30.

Dagan, I., L. Lee, and F. Pereira. 1997. Similarity-Based Methods for Word Sense Disambiguation. *Thirty-Fifth Annual Meeting of the ACL and Eighth Conference of the EACL*, pages 56–63.

Davel, M. and E. Barnard. 2008. Pronunciation prediction with default refine. *Computer Speech and Language*, 22:374–393.

Erjavec, T. and S. Dzeroski. 2004. Machine learning of morphosyntactic structure: Lemmatising unknown Slovene words. *Applied Artificial Intelligence*, 18(1):17–40.

Goldsmith, J. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27:153–198.

Golénia, B., S. Spiegler, and P. Flach. 2009. UNGRADE: UNsupervised GRAph DEcomposition. In *Working Notes for the CLEF 2009 Workshop, CLEF 2009, Corfu, Greece*.

Harris, Z. 1955. From Phoneme to Morpheme. *Language*, 31(2):190–222.

Kazakov, D. and S. Manandhar. 2001. Unsupervised learning of word segmentation rules with genetic algorithms and inductive logic programming. *Machine Learning*, 43:121–162.

Koskenniemmi, K. 1983. *Two-level Morphology: A General Computational Model for Word Form Recognition and Production*. Ph.D. thesis, University of Helsinki.

Li, W. 2001. New stopping criteria for segmenting DNA sequences. *Physical Review Letters*, 86(25):5815–5818.

Oflazer, K., M. McShane, and S. Nirenburg. 2001. Bootstrapping morphological analyzers by combining human elicitation and machine learning. *Computational Linguistics*, 27(1):59–85.

Shalonova, K., B. Golenia, and P. Flach. 2009. Towards learning morphology for under-resourced languages. *IEEE Transactions on Audio, Speech and Language Procesing*, 17(5):956–965.

Spiegler, S., B. Golenia, K. Shalonova, P. Flach, and R. Tucker. 2008. Learning the morphology of Zulu with different degrees of supervision. *IEEE Spoken Language Technology Workshop*, pages 9–12.

Torkkola, K. 1993. An efficient way to learn English grapheme-to-phoneme rules automatically. *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 199–202.

Multi-Document Summarization via the Minimum Dominating Set

Chao Shen and **Tao Li**
School of Computing and Information Sciences
Florida Internation University
{cshen001|taoli}@cs.fiu.edu

Abstract

Multi-document summarization has been an important problem in information retrieval. It aims to distill the most important information from a set of documents to generate a compressed summary. Given a sentence graph generated from a set of documents where vertices represent sentences and edges indicate that the corresponding vertices are similar, the extracted summary can be described using the idea of graph domination. In this paper, we propose a new principled and versatile framework for multi-document summarization using the minimum dominating set. We show that four well-known summarization tasks including generic, query-focused, update, and comparative summarization can be modeled as different variations derived from the proposed framework. Approximation algorithms for performing summarization are also proposed and empirical experiments are conducted to demonstrate the effectiveness of our proposed framework.

1 Introduction

As a fundamental and effective tool for document understanding and organization, multi-document summarization enables better information services by creating concise and informative reports for a large collection of documents. Specifically, in multi-document summarization, given a set of documents as input, the goal is to produce a condensation (i.e., a generated summary) of the content of the entire input set (Jurafsky and Martin, 2008). The generated summary can be generic where it simply gives the important information contained in the input documents without any particular information needs or query/topic-focused where it is produced in response to a user query or related to a topic or concern the development of an event (Jurafsky and Martin, 2008; Mani, 2001).

Recently, new summarization tasks such as update summarization (Dang and Owczarzak, 2008) and comparative summarization (Wang et al., 2009a) have also been proposed. Update summarization aims to generate short summaries of recent documents to capture new information different from earlier documents and comparative summarization aims to summarize the differences between comparable document groups.

In this paper, we propose a new principled and versatile framework for multi-document summarization using the *minimum dominating set*. Many known summarization tasks including generic, query-focused, update, and comparative summarization can be modeled as different variations derived from the proposed framework. The framework provides an elegant basis to establish the connections between various summarization tasks while highlighting their differences.

In our framework, a sentence graph is first generated from the input documents where vertices represent sentences and edges indicate that the corresponding vertices are similar. A natural method for describing the extracted summary is based on the idea of graph domination (Wu and Li, 2001). *A dominating set* of a graph is a subset of vertices such that every vertex in the graph is either in the subset or adjacent to a vertex in the subset; and

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 984–992,
Beijing, August 2010

a minimum dominating set is a dominating set with the minimum size. The minimum dominating set of the sentence graph can be naturally used to describe the summary: it is *representative* since each sentence is either in the minimum dominating set or connected to one sentence in the set; and it is with *minimal redundancy* since the set is of minimum size. Approximation algorithms are proposed for performing summarization and empirical experiments are conducted to demonstrate the effectiveness of our proposed framework. Though the dominating set problem has been widely used in wireless networks, this paper is the first work on using it for modeling sentence extraction in document summarization.

The rest of the paper is organized as follows. In Section 2, we review the related work about multi-document summarization and the dominating set. After introducing the minimum dominating set problem in graph theory in Section 3, we propose the minimum dominating set based framework for multi-document summarization and model the four summarization tasks including generic, query-focused, update, and comparative summarization in Section 4. Section 5 presents the experimental results and analysis, and finally Section 6 concludes the paper.

2 Related Work

Generic Summarization For generic summarization, a saliency score is usually assigned to each sentence and then the sentences are ranked according to the saliency score. The scores are usually computed based on a combination of statistical and linguistic features. MEAD (Radev et al., 2004) is an implementation of the centroid-based method where the sentence scores are computed based on sentence-level and inter-sentence features. SumBasic (Nenkova and Vanderwende, 2005) shows that the frequency of content words alone can also lead good summarization results. Graph-based methods (Erkan and Radev, 2004; Wan et al., 2007b) have also been proposed to rank sentences or passages based on the PageRank algorithm or its variants.

Query-Focused Summarization In query-focused summarization, the information of the given topic or query should be incorporated into summarizers, and sentences suiting the user's declared information need should be extracted. Many methods for generic summarization can be extended to incorporate the query information (Saggion et al., 2003; Wei et al., 2008). Wan et al. (Wan et al., 2007a) make full use of both the relationships among all the sentences in the documents and relationship between the given query and the sentences by manifold ranking. Probability models have also been proposed with different assumptions on the generation process of the documents and the queries (Daumé III and Marcu, 2006; Haghighi and Vanderwende, 2009; Tang et al., 2009).

Update Summarization and Comparative Summarization Update summarization was introduced in Document Understanding Conference (DUC) 2007 (Dang, 2007) and was a main task of the summarization track in Text Analysis Conference (TAC) 2008 (Dang and Owczarzak, 2008). It is required to summarize a set of documents under the assumption that the reader has already read and summarized the first set of documents as the main summary. To produce the update summary, some strategies are required to avoid redundant information which has already been covered by the main summary. One of the most frequently used methods for removing redundancy is Maximal Marginal Relevance(MMR) (Goldstein et al., 2000). Comparative document summarization is proposed by Wang et. al. (Wang et al., 2009a) to summarize the differences between comparable document groups. A sentence selection approach is proposed in (Wang et al., 2009a) to accurately discriminate the documents in different groups modeled by the conditional entropy.

The Dominating Set Many approximation algorithms have been developed for finding minimum dominating set for a given graph (Guha and Khuller, 1998; Thai et al., 2007). Kann (Kann, 1992) shows that the minimum dominating set problem is equivalent to set cover problem, which is a well-known NP-hard problem. Dominating set has been widely used for clustering in wireless networks (Chen and Liestman, 2002; Han and Jia, 2007). It has been used to find topic words for hierarchical summarization (Lawrie et al., 2001), where a set of topic words is extracted as a dominating set of word graph. In our work, we use the minimum dominating set to formalize the sentence extraction for document summarization.

3 The Minimum Dominating Set Problem

Given a graph $G =< V, E >$, *a dominating set* of G is a subset S of vertices with the following property: each vertex of G is either in the dominating set S, or is adjacent to some vertices in S.

Problem 3.1. *Given a graph G, the minimum dominating set problem (MDS) is to find a minimum size subset S of vertices, such that S forms a dominating set.*

MDS is closely related to the set cover problem (SC), a well-known NP-hard problem.

Problem 3.2. *Given F, a finite collection $\{S_1, S_2, \ldots, S_n\}$ of finite sets, the set cover problem (SC) is to find the optimal solution*

$$F^* = \arg\min_{F' \subseteq F} |F'| \ s.t. \bigcup_{S' \in F'} S' = \bigcup_{S \in F} S.$$

Theorem 3.3. *There exists a pair of polynomial time reduction between MDS and SC.*

So, MDS is also NP-hard and it has been shown that there are no approximate solutions within $c \log |V|$, for some $c > 0$ (Feige, 1998; Raz and Safra, 1997).

3.1 An Approximation Algorithm

A greedy approximation algorithm for the SC problem is described in (Johnson, 1973). Basically, at each stage, the greedy algorithm chooses the set which contains the largest number of uncovered elements.

Based on Theorem 3.3, we can obtain a greedy approximation algorithm for MDS. Starting from an empty set, if the current subset of vertices is not the dominating set, a new vertex which has the most number of the adjacent vertices that are not adjacent to any vertex in the current set will be added.

Proposition 3.4. *The greedy algorithm approximates SC within $1 + \ln s$ where s is the size of the largest set.*

It was shown in (Johnson, 1973) that the approximation factor for the greedy algorithm is no more than $H(s)$, the s-th harmonic number:

$$H(s) = \sum_{k=1}^{s} \frac{1}{k} \leq \ln s + 1$$

Corollary 3.5. *MDS has a approximation algorithm within $1 + \ln \Delta$ where Δ is the maximum degree of the graph.*

Corollary 3.5 follows directly from Theorem 3.3 and Proposition 3.4.

4 The Summarization Framework

4.1 Sentence Graph Generation

To perform multi-document summarization via minimum dominating set, we need to first construct a sentence graph in which each node is a sentence in the document collection. In our work, we represent the sentences as vectors based on tf-isf, and then obtain the cosine similarity for each pair of sentences. If the similarity between a pair of sentences s_i and s_j is above a given threshold λ, then there is an edge between s_i and s_j.

For generic summarization, we use all sentences for building the sentence graph. For query-focused summarization, we only use the sentences containing at least one term in the query. In addition, when a query q is involved, we assign each node s_i a weight, $w(s_i) = d(s_i, q) = 1 - \cos(s_i, q)$, to indicate the distance between the sentence and the query q.

After building the sentence graph, we can formulate the summarization problem using

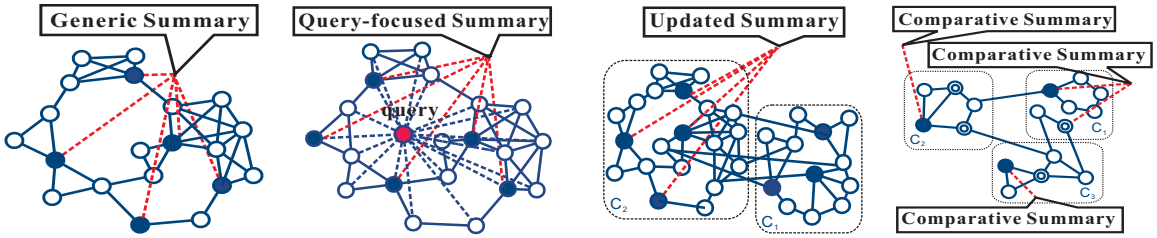


Figure 1: Graphical illustrations of multi-document summarization via the minimum dominating set. (a): The minimum dominating set is extracted as the generic summary. (b):The minimum weighted dominating set is extracted as the query-based summary. (c):Vertices in the right rectangle represent the first document set C_1, and ones in the left represent the second document set where update summary is generated. (d):Each rectangle represents a group of documents. The vertices with rings are the dominating set for each group, while the solid vertices are the complementary dominating set, which is extracted as comparative summaries.

the minimum dominating set. A graphical illustration of the proposed framework is shown in Figure 1.

4.2 Generic Summarization

Generic summarization is to extract the most representative sentences to capture the important content of the input documents. Without taking into account the length limitation of the summary, we can assume that the summary should represent all the sentences in the document set (i.e., every sentence in the document set should either be extracted or be similar with one extracted sentence). Meanwhile, a summary should also be as short as possible. Such summary of the input documents under the assumption is exactly the minimum dominating set of the sentence graph we constructed from the input documents in Section 4.1. Therefore the summarization problem can be formulated as the minimum dominating set problem.

However, usually there is a length restriction for generating the summary. Moreover, the MDS is NP-hard as shown in Section 3. Therefore, it is straightforward to use a greedy approximation algorithm to construct a subset of the dominating set as the final summary. In the greedy approach, at each stage, a sentence which is optimal according to the local criteria will be extracted. Algorithm 1 describes

Algorithm 1 Algorithm for Generic Summarization

INPUT: G, W
OUTPUT: S

$S = \emptyset$
$T = \emptyset$
while $L(S) < W$ and $V(G)! = S$ **do**
  for $v \in V(G) - S$ **do**
    $s(v) = |\{ADJ(v) - T\}|$
  $v^* = \arg\max_v s(v)$
  $S = S \cup \{v^*\}$
  $T = T \cup ADJ(v^*)$

an approximation algorithm for generic summarization. In Algorithm 1, G is the sentence graph, $L(S)$ is the length of the summary, W is the maximal length of the summary, and $ADJ(v) = \{v' | (v', v) \in E(G)\}$ is the set of vertices which are adjacent to the vertex v. A graphical illustration of generic summarization using the minimum dominating set is shown in Figure 1(a).

4.3 Query-Focused Summarization

Letting G be the sentence graph constructed in Section 4.1 and q be the query, the query-focused summarization can be modeled as

$$D^* = \arg\min_{D \subseteq G} \sum_{s \in D} d(s, q) \quad (1)$$
$$\text{s.t. } D \text{ is a dominating set of } G.$$

Note that $d(s, q)$ can be viewed as the weight of vertex in G. Here the summary length is minimized implicitly, since if $D' \subseteq D$, then

$\sum_{s\in D'} d(s,q) \leq \sum_{s\in D} d(s,q)$. The problem in Eq.(1) is exactly a variant of the minimum dominating set problem, i.e., the minimum weighted dominating set problem (MWDS).

Similar to MDS, MWDS can be reduced from the weighted version of the SC problem. In the weighted version of SC, each set has a weight and the sum of weights of selected sets needs to be minimized. To generate an approximate solution for the weighted SC problem, instead of choosing a set i maximizing $|SET(i)|$, a set i minimizing $\frac{w(i)}{|SET(i)|}$ is chosen, where $SET(i)$ is composed of uncovered elements in set i, and $w(i)$ is the weight of set i. The approximate solution has the same approximation ratio as that for MDS, as stated by the following theorem (Chvatal, 1979).

Theorem 4.1. *An approximate weighted dominating set can be generated with a size at most* $1+\log\Delta\cdot|OPT|$*, where* Δ *is the maximal degree of the graph and* OPT *is the optimal weighted dominating set.*

Accordingly, from generic summarization to query-focused summarization, we just need to modify line 6 in Algorithm 1 to

$$v^* = \arg\min_v \frac{w(v)}{s(v)}, \tag{2}$$

where $w(v)$ is the weight of vertex v. A graphical illustration of query-focused summarization using the minimum dominating set is shown in Figure 1(b).

4.4 Update Summarization

Give a query q and two sets of documents C_1 and C_2, update summarization is to generate a summary of C_2 based on q, given C_1. Firstly, summary of C_1, referred as D_1 can be generated. Then, to generate the update summary of C_2, referred as D_2, we assume D_1 and D_2 should represent all query related sentences in C_2, and length of D_2 should be minimized.

Let G_1 be the sentence graph for C_1. First we use the method described in Section 4.3 to extract sentences from G_1 to form D_1. Then we expand G_1 to the whole graph G using the second set of documents C_2. G is then the graph presentation of the document set including C_1 and C_2. We can model the update summary of C_2 as

$$D^* = \arg\min_{D_2} \sum_{s\in D_2} w(s) \tag{3}$$
$$\text{s.t. } D_2 \cup D_1 \text{ is a dominating set of } G.$$

Intuitively, we extract the smallest set of sentences that are closely related to the query from C_2 to complete the partial dominating set of G generated from D_1. A graphical illustration of update summarization using the minimum dominating set is shown in Figure 1(c).

4.5 Comparative Summarization

Comparative document summarization aims to summarize the differences among comparable document groups. The summary produced for each group should emphasize its difference from other groups (Wang et al., 2009a).

We extend our method for update summarization to generate the discriminant summary for each group of documents. Given N groups of documents $C_1, C_2, \ldots, C_N$, we first generate the sentence graphs $G_1, G_2, \ldots, G_N$, respectively. To generate the summary for $C_i, 1 \leq i \leq N$, we view C_i as the update of all other groups. To extract a new sentence, only the one connected with the largest number of sentences which have no representatives in any groups will be extracted. We denote the extracted set as the complementary dominating set, since for each group we obtain a subset of vertices dominating those are not dominated by the dominating sets of other groups. To perform comparative summarization, we first extract the standard dominating sets for $G_1, \ldots, G_N$, respectively, denoted as $D_1, \ldots, D_N$. Then we extract the so-called complementary dominating set CD_i for G_i by continuing adding vertices in G_i to find the dominating set of $\cup_{1\leq j\leq N} G_j$ given $D_1, \ldots, D_{i-1}, D_{i+1}, \ldots, D_N$. A graphical illustration of comparative summarization is shown in Figure 1(d).

	DUC04	DUC05	DUC06	TAC08 A	TAC08 B
Type of Summarization	Generic	Topic-focused	Topic-focused	Topic-focused	Update
#topics	NA	50	50	48	48
#documents per topic	10	25-50	25	10	10
Summary length	665 bytes	250 words	250 words	100 words	100 words

Table 1: Brief description of the data set

5 Experiments

We have conducted experiments on all four summarization tasks and our proposed methods based on the minimum dominating set have outperformed many existing methods. For the generic, topic-focused and update summarization tasks, the experiments are perform the DUC data sets using ROUGE-2 and ROUGE-SU (Lin and Hovy, 2003) as evaluation measures. For comparative summarization, a case study as in (Wang et al., 2009a) is performed. Table 1 shows the characteristics of the data sets. We use DUC04 data set to evaluate our method for generic summarization task and DUC05 and DUC06 data sets for query-focused summarization task. The data set for update summarization, (i.e. the main task of TAC 2008 summarization track) consists of 48 topics and 20 newswire articles for each topic. The 20 articles are grouped into two clusters. The task requires to produce 2 summaries, including the initial summary (TAC08 A) which is standard query-focused summarization and the update summary (TAC08 B) under the assumption that the reader has already read the first 10 documents.

We apply a 5-fold cross-validation procedure to choose the threshold λ used for generating the sentence graph in our method.

5.1 Generic Summarization

We implement the following widely used or recent published methods for generic summarization as the baseline systems to compare with our proposed method (denoted as MDS). (1) Centroid: The method applies MEAD algorithm (Radev et al., 2004) to extract sentences according to the following three parameters: centroid value, positional value, and first-sentence overlap. (2) LexPageRank: The method first constructs a sentence connectivity graph based on cosine similarity and then selects important sentences based on the concept of eigenvector centrality (Erkan and Radev, 2004). (3) BSTM: A Bayesian sentence-based topic model making use of both the term-document and term-sentence associations (Wang et al., 2009b).

Our method outperforms the simple Centroid method and another graph-based LexPageRank, and its performance is close to the results of the Bayesian sentence-based topic model and those of the best team in the DUC competition. Note however that, like clustering or topic based methods, BSTM needs the topic number as the input, which usually varies by different summarization tasks and is hard to estimate.

5.2 Query-Focused Summarization

We compare our method (denoted as MWDS) described in Section 4.3 with some recently published systems. (1) TMR (Tang et al., 2009): incorporates the query information into the topic model, and uses topic based score and term frequency to estimate the importance of the sentences. (2) SNMF (Wang et al., 2008): calculates sentence-sentence similarities by sentence-level semantic analysis, clusters the sentences via symmetric nonnegative matrix factorization, and extracts the sentences based on the clustering result. (3) Wiki (Nastase, 2008): uses Wikipedia as external knowledge to expand query and builds the connection between the query and the sentences in documents.

Table 3 presents the experimental comparison of query-focused summarization on the two datasets. From Table 3, we observe that our method is comparable with these systems. This is due to the good interpretation of the summary extracted by our method, an ap-

	DUC04	
	ROUGE-2	ROUGE-SU
DUC Best	0.09216	0.13233
Centroid	0.07379	0.12511
LexPageRank	0.08572	0.13097
BSTM	0.09010	0.13218
MDS	0.08934	0.13137

Table 2: Results on generic summarization.

	DUC05		DUC06	
	ROUGE-2	ROUGE-SU	ROUGE-2	ROUGE-SU
DUC Best	0.0725	0.1316	0.09510	0.15470
SNMF	0.06043	0.12298	0.08549	0.13981
TMR	0.07147	0.13038	0.09132	0.15037
Wiki	0.07074	0.13002	0.08091	0.14022
MWDS	0.07311	0.13061	0.09296	0.14797

Table 3: Results on query-focused summarization.

proximate minimal dominating set of the sentence graph. On DUC05, our method achieves the best result; and on DUC06, our method outperforms all other systems except the best team in DUC. Note that our method based on the minimum dominating set is much simpler than other systems. Our method only depends on the distance to the query and has only one parameter (i.e., the threshold λ in generating the sentence graph).

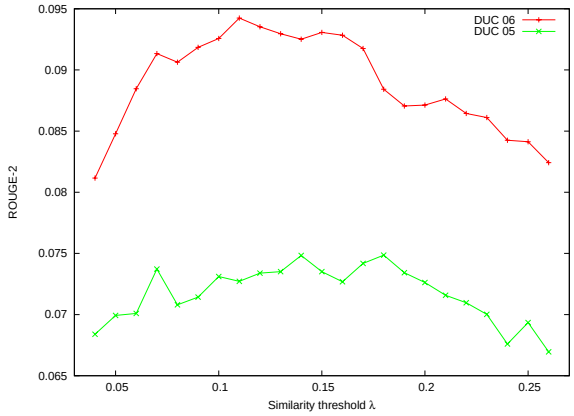


Figure 2: ROUGE-2 vs. threshold λ

We also conduct experiments to empirically evaluate the sensitivity of the threshold λ. Figure 2 shows the ROUGE-2 curve of our MWDS method on the two datasets when λ varies from 0.04 to 0.26. When λ is small, edges fail to represent the similarity of the sentences, while if λ is too large, the graph will be sparse. As λ is approximately in the range of $0.1 - 0.17$, ROUGE-2 value becomes stable and relatively high.

5.3 Update Summarization

Table 5 presents the experimental results on update summarization. In Table 5, 'TAC Best" and "TAC Median" represent the best and median results from the participants of TAC 2008 summarization track in the two tasks respectively according to the TAC 2008 report (Dang and Owczarzak, 2008). As seen from the results, the ROUGE scores of our methods are higher than the median results. The good results of the best team typically come from the fact that they utilize advanced natural language processing (NLP) techniques to resolve pronouns and other anaphoric expressions. Although we can spend more efforts on the preprocessing or language processing step, our goal here is to demonstrate the effectiveness of formalizing the update summarization problem using the minimum dominating set and hence we do not utilize advanced NLP techniques for preprocessing. The experimental results demonstrate that our simple update summarization method based on the minimum dominating set can lead to competitive performance for update summarization.

	TAC08 A		TAC08 B	
	ROUGE-2	ROUGE-SU	ROUGE-2	ROUGE-SU
TAC Best	0.1114	0.14298	0.10108	0.13669
TAC Median	0.08123	0.11975	0.06927	0.11046
MWDS	0.09012	0.12094	0.08117	0.11728

Table 5: Results on update summarization.

5.4 Comparative Summarization

We use the top six largest clusters of documents from TDT2 corpora to compare the summary generated by different comparative summarization methods. The topics of the six document clusters are as follows: topic 1: Iraq Issues; topic 2: Asia's economic crisis; topic 3: Lewinsky scandal; topic 4: Nagano Olympic Games; topic 5: Nuclear Issues in Indian and Pakistan; and topic 6: Jakarta Riot. From each of the topics, 30 documents are extracted

Topic	Complementary Dominating Set	Discriminative Sentence Selection	Dominating Set
1	··· U.S. Secretary of State Madeleine Albright arrives to consult on the stand-off between the **United Nations and Iraq**.	**the U.S. envoy to the United Nations**, Bill Richardson, ··· play down China's refusal to support threats of **military force against Iraq**	The United States and Britain do not trust **President Saddam** and wants *cdots*warning of serious consequences if **Iraq** violates the accord.
2	**Thailand's currency**, the baht, **dropped through a key psychological level** of ··· amid a regional sell-off sparked by escalating **social unrest in Indonesia**.	Earlier, driven largely by **the declining yen**, **South Korea's stock market fell** by ···, while the **Nikkei 225 benchmark index dipped** below 15,000 in the morning ···	*In the fourth quarter, IBM Corp. earned $2.1 billion, up 3.4 percent from $2 billion a year earlier.*
3	··· attorneys representing **President Clinton and Monica Lewinsky.**	The following night **Isikoff** ···, where he directly followed the recitation of the top-10 list: "Top 10 **White House Jobs That Sound Dirty.**"	In Washington, Ken Starr's grand jury continued its investigation of the **Monica Lewinsky matter**.
4	Eight women and six men were named Saturday night as the first **U.S. Olympic Snowboard Team** as their sport gets set to make its debut in **Nagano, Japan**.	*this tunnel is finland's cross country version of tokyo's alpine ski dome, and olympic skiers flock from russia, ···, france and austria this past summer to work out the kinks* ···	If the skiers the **men's super-G** and the **women's downhill** on Saturday, they will be back on schedule.
5	**U.S. officials** have announced **sanctions** Washington will impose on **India and Pakistan** for conducting **nuclear tests**.	The **sanctions** would stop all foreign aid except for humanitarian purposes, **ban military sales to India** ···	And **Pakistan's prime minister** says his country will sign the **U.N.'s comprehensive ban on nuclear tests** if **India** does, too.
6	··· remain in force around **Jakarta**, and at the Parliament building where **thousands of students staged a sit-in** Tuesday ···.	"**President Suharto** has given much to his country over the past 30 years, raising **Indonesia's** standing in the world ···	*What were the students doing at the time you were there, and what was the reaction of the students to the troops?*

Table 4: A case study on comparative document summarization. Some unimportant words are skipped due to the space limit. The bold font is used to annotate the phrases that are highly related with the topics, and italic font is used to highlight the sentences that are not proper to be used in the summary.

randomly to produce a one-sentence summary. For comparison purpose, we extract the sentence with the maximal degree as the baseline. Note that the baseline can be thought as an approximation of the dominating set using only one sentence. Table 4 shows the summaries generated by our method (complementary dominating set (CDS)), discriminative sentence selection (DSS) (Wang et al., 2009a) and the baseline method. Our CDS method can extract discriminative sentences for all the topics. DSS can extract discriminative sentences for all the topics except topic 4. Note that the sentence extracted by DSS for topic 4 may be discriminative from other topics, but it is deviated from the topic Nagano Olympic Games. In addition, DSS tends to select long sentences which should not be preferred for summarization purpose. The baseline method may extract some general sentences, such as the sentence for topic 2 and topic 6 in Table 4.

6 Conclusion

In this paper, we propose a framework to model the multi-document summarization using the minimum dominating set and show that many well-known summarization tasks can be formulated using the proposed framework. The proposed framework leads to simple yet effective summarization methods. Experimental results show that our proposed methods achieve good performance on several multi-document document tasks.

7 Acknowledgements

This work is supported by NSF grants IIS-0549280 and HRD-0833093.

References

Chen, Y.P. and A.L. Liestman. 2002. Approximating minimum size weakly-connected dominating sets for clustering mobile ad hoc networks. In *Proceedings of International Symposium on Mobile Ad hoc Networking & Computing*. ACM.

Chvatal, V. 1979. A greedy heuristic for the set-covering problem. *Mathematics of operations research*, 4(3):233–235.

Dang, H.T. and K Owczarzak. 2008. Overview of the TAC 2008 Update Summarization Task. In *Proceedings of the Text Analysis Conference (TAC)*.

Dang, H.T. 2007. Overview of DUC 2007. In *Document Understanding Conference*.

Daumé III, H. and D. Marcu. 2006. Bayesian query-focused summarization. In *Proceedings of the ACL-COLING*.

Erkan, G. and D.R. Radev. 2004. Lexpagerank: Prestige in multi-document text summarization. In *Proceedings of EMNLP*.

Feige, U. 1998. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652.

Goldstein, J., V. Mittal, J. Carbonell, and M. Kantrowitz. 2000. Multi-document summarization by sentence extraction. In *NAACL-ANLP 2000 Workshop on Automatic summarization*.

Guha, S. and S. Khuller. 1998. Approximation algorithms for connected dominating sets. *Algorithmica*, 20(4):374–387.

Haghighi, A. and L. Vanderwende. 2009. Exploring content models for multi-document summarization. In *Proceedings of HLT-NAACL*.

Han, B. and W. Jia. 2007. Clustering wireless ad hoc networks with weakly connected dominating set. *Journal of Parallel and Distributed Computing*, 67(6):727–737.

Johnson, D.S. 1973. Approximation algorithms for combinatorial problems. In *Proceedings of STOC*.

Jurafsky, D. and J.H. Martin. 2008. *Speech and language processing*. Prentice Hall New York.

Kann, V. 1992. *On the approximability of NP-complete optimization problems*. PhD thesis, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm.

Lawrie, D., W.B. Croft, and A. Rosenberg. 2001. Finding topic words for hierarchical summarization. In *Proceedings of SIGIR*.

Lin, C.Y. and E. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of HLT-NAACL*.

Mani, I. 2001. Automatic summarization. *Computational Linguistics*, 28(2).

Nastase, V. 2008. Topic-driven multi-document summarization with encyclopedic knowledge and spreading activation. In *Proceedings of EMNLP*.

Nenkova, A. and L. Vanderwende. 2005. The impact of frequency on summarization. *Microsoft Research, Redmond, Washington, Tech. Rep. MSR-TR-2005-101*.

Radev, D.R., H. Jing, M. Styś, and D. Tam. 2004. Centroid-based summarization of multiple documents. *Information Processing and Management*, 40(6):919–938.

Raz, R. and S. Safra. 1997. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of STOC*.

Saggion, H., K. Bontcheva, and H. Cunningham. 2003. Robust generic and query-based summarisation. In *Proceedings of EACL*.

Tang, J., L. Yao, and D. Chen. 2009. Multi-topic based Query-oriented Summarization. In *Proceedings of SDM*.

Thai, M.T., N. Zhang, R. Tiwari, and X. Xu. 2007. On approximation algorithms of k-connected m-dominating sets in disk graphs. *Theoretical Computer Science*, 385(1-3):49–59.

Wan, X., J. Yang, and J. Xiao. 2007a. Manifold-ranking based topic-focused multi-document summarization. In *Proceedings of IJCAI*.

Wan, X., J. Yang, and J. Xiao. 2007b. Towards an iterative reinforcement approach for simultaneous document summarization and keyword extraction. In *Proceedings of ACL*.

Wang, D., T. Li, S. Zhu, and C. Ding. 2008. Multi-document summarization via sentence-level semantic analysis and symmetric matrix factorization. In *Proceedings of SIGIR*.

Wang, D., S. Zhu, T. Li, and Y. Gong. 2009a. Comparative document summarization via discriminative sentence selection. In *Proceeding of CIKM*.

Wang, D., S. Zhu, T. Li, and Y. Gong. 2009b. Multi-document summarization using sentence-based topic models. In *Proceedings of the ACL-IJCNLP*.

Wei, F., W. Li, Q. Lu, and Y. He. 2008. Query-sensitive mutual reinforcement chain and its application in query-oriented multi-document summarization. In *Proceedings of SIGIR*.

Wu, J. and H. Li. 2001. A dominating-set-based routing scheme in ad hoc wireless networks. *Telecommunication Systems*, 18(1):13–36.

Corpus-based Semantic Class Mining: Distributional vs. Pattern-Based Approaches

Shuming Shi[1] **Huibin Zhang**[2*] **Xiaojie Yuan**[2] **Ji-Rong Wen**[1]
[1] Microsoft Research Asia
[2] Nankai University
{shumings, jrwen}@microsoft.com
zhanghuibin@126.com; yuanxj@nankai.edu.cn

Abstract

Main approaches to corpus-based semantic class mining include *distributional similarity* (DS) and *pattern-based* (PB). In this paper, we perform an empirical comparison of them, based on a publicly available dataset containing 500 million web pages, using various categories of queries. We further propose a frequency-based rule to select appropriate approaches for different types of terms.

1 Introduction

Computing the semantic relationship between terms, which has wide applications in natural language processing and web search, has been a hot topic nowadays. This paper focuses on corpus-based semantic class mining (Lin 1998; Pantel and Lin 2002; Pasca 2004; Shinzato and Torisawa, 2005; Ohshima, et al., 2006; Zhang et al., 2009), where *peer terms* (or *coordinate terms*) are discovered from a corpus.

Existing approaches to semantic class mining could roughly be divided into two categories: *distributional similarity* (*DS*), and *pattern-based* (*PB*). The first type of work (Hindle, 1990; Lin 1998; Pantel and Lin 2002) is based on the distributional hypothesis (Harris, 1985), saying that terms occurring in analogous (lexical or syntactic) *contexts* tend to be similar. DS approaches basically exploit *second-order* co-occurrences to discover strongly associated concepts. In pattern-based approaches (Hearst 1992; Pasca 2004; Shinzato and Torisawa, 2005; Ohshima, et al., 2006; Zhang et al., 2009), patterns are applied to discover specific relationships between terms, from the general *first-order* co-occurrences. For example, "*NP such as NP, NP…, and NP*" is a popular and high-quality pattern for extracting peer terms (and also hyponyms). Besides the natural language patterns, some HTML tag tree patterns (e.g., the drop down list) are also effective in semantic class mining.

It is worth-noting that the word "*pattern*" also appears in some DS approaches (Pasca et al., 2006; Tanev and Magnini, 2006; Pennacchiotti and Pantel, 2009), to represent the *context* of a term or a term-pair, e.g., "(invent, subject-of)" for the term "Edison", and "- starring -" for the term-pair "(The Terminal, Tom Hanks)". Although "*patterns*" are utilized, we categorize them as DS approaches rather than PB, because they match the DS framework well. In this paper, *PB* only refers to the approaches that utilize patterns to exploit *first-order* co-occurrences. And the patterns in DS approaches are called *contexts* in the following part of this paper.

Progress has been made and promising results have been reported in the past years for both DS and PB approaches. However, most previous research work (some exceptions are discussed in related work) involves solely one category of approach. And there is little work studying the comparison of their performance for different types of terms (we use "*term*" to represent a single word or a phrase).

In this paper, we make an empirical study of this problem, based on a large-scale, publicly available dataset containing 500 million web pages. For each approach *P*, we build a *term-similarity graph G*(*P*), with vertices representing terms, and edges being the confidence that the two terms are peers. Approaches are compared by the quality of their corresponding term graphs.

[*] Work done during an internship at Microsoft

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 993–1001,
Beijing, August 2010

We measure the quality of a term graph by set expansion. Two query sets are adopted: One contains 49 semantic classes of named entities and 20220 trials (queries), collected by Pantel et al. (2009) from Wikipedia[2]; and the other contains 100 queries of five lexical categories (proper nouns, common nouns, verbs, adjectives, and adverbs), built in this paper for studying the performance comparison on different term types. With the dataset and the query sets, we study the comparison of DS and PB. Key observations and preliminary conclusions are,

- ***DS vs. PB***: DS approaches perform much better on common nouns, verbs, adjectives, and adverbs; while PB generates higher-quality semantic classes for proper nouns.
- ***Lexical vs. Html-tag patterns***: If only lexical patterns are adopted in PB, the performance drops significantly; while the performance only becomes slightly worse with only Html-tag patterns being included.
- ***Corpus-size***: For proper nouns, PB beats DS even based on a much smaller corpus; similarly, for other term types, DS performs better even with a smaller corpus.

Given these observations, we further study the feasibility of selecting appropriate approaches for different term types to obtain better results. A simple and effective frequency-based rule is proposed for approach-selection. Our online semantic mining system (*NeedleSeek*)[3] adopts both PB and DS to build semantic classes.

2 Related Work

Existing efforts for semantic class mining has been done upon various types of data sources, including text-corpora, search-results, and query logs. In corpus-based approaches (Lin 1998; Lin and Pantel 2001; Pantel and Lin 2002; Pasca 2004; Zhang et al., 2009), semantic classes are obtained by the offline processing of a corpus which can be unstructured (e.g., plain text) or semi-structured (e.g., web pages). Search-results-based approaches (Etzioni et al., 2004; Kozareva et al., 2008; Wang and Cohen, 2008) assume that multiple terms (or, less often, one term) in a semantic class have been provided as *seeds*. Other terms in the class are retrieved by sending queries (constructed according to the seeds) to a web search engine and mining the search results. Query logs are exploited in (Pasca 2007; Komachi and Suzuki, 2008; Yamaguchi 2008) for semantic class mining. This paper focuses on corpus-based approaches.

As has been mentioned in the introduction part, primarily two types of methodologies are adopted: DS and PB. Syntactic context information is used in (Hindle, 1990; Ruge, 1992; Lin 1998; Lin and Pantel, 2001; Pantel and Lin, 2002) to compute term similarities. The construction of syntactic contexts requires sentences to be parsed by a dependency parser, which may be extremely time-consuming on large corpora. As an alternative, lexical context (such as text window) has been studied (Pantel et al., 2004; Agirre et al., 2009; Pantel et al., 2009). In the pattern-based category, a lot of work has been done to discover term relations by sentence lexical patterns (Hearst 1992; Pasca 2004), HTML tag patterns (Shinzato and Torisawa, 2005), or both (Shi et al., 2008; Zhang et al., 2009). In this paper, our focus is not one specific methodology, but the comparison and combination of them.

A small amount of existing work is related to the comparison or combination of multiple methods. Pennacchiotti and Pantel (2009) proposed a feature combination framework (named ensemble semantic) to combine features generated by different extractors (distributional and "pattern-based") from various data sources. As has been discussed in the introduction, in our terminology, their "pattern-based" approaches are actually DS for term-pairs. In addition, their study is based on three semantic classes (actors, athletes, and musicians), all of which are proper nouns. Differently, we perform the comparison by classifying terms according to their lexical categories, based on which additional insights are obtained about the pros and cons of each methodology. Pantel et al., (2004) proposed, in the scenario of extracting *is-a* relations, one pattern-based approach and compared it with a baseline syntactic distributional similarity method (called *syntactic co-occurrence* in their paper). Differently, we study the comparison in a different scenario (semantic class mining). In addition, they did not differentiate the lexical types of terms in the study. The third difference is that we proposed a rule for method-selection while they did not. In (Pasca and Durme,

[2] http://www.wikipedia.org/
[3] http://needleseek.msra.cn/

2008), clusters of distributional similar terms were adopted to expand the labeled semantic classes acquired from the "such as | including" pattern. Although both patterns and distributional similarity were used in their paper, they did not do any comparison about their performance. Agirre et al. (2009) compared DS approaches with WordNet-based methods in computing word similarity and relatedness; and they also studied the combination of them. Differently, the methods for comparison in our paper are DS and PB.

3 Similarity Graph Construction

A key operation in corpus-based semantic class mining is to build a term similarity graph, with vertices representing terms, and edges being the similarity (or distance) between terms. Given the graph, a clustering algorithm can be adopted to generate the final semantic classes. Now we describe the state-of-the-art DS and PB approaches for computing term similarities.

3.1 Distributional Similarity

DS approaches are based on the distributional hypothesis (Harris, 1985), which says that terms appearing in analogous contexts tend to be similar. In a DS approach, a term is represented by a feature vector, with each feature corresponding to a context in which the term appears. The similarity between two terms is computed as the similarity between their corresponding feature vectors. Different approaches may have different ways of 1) defining a context, 2) assigning feature values, or 3) measuring the similarity between two feature vectors.

Contexts	Text window (window size: 2, 4)
	Syntactic
Feature value	PMI
Similarity measure	Cosine, Jaccard

Table 1. DS approaches implemented in this paper

Mainly two kinds of contexts have been extensively studied: *syntactic context* and *lexical context*. The construction of *syntactic contexts* relies on the syntactic parsing trees of sentences, which are typically the output of a syntactic parser. Given a syntactic tree, a syntactic context of a term w can be defined as the parent (or one child) of w in the tree together with their relationship (Lin, 1998; Pantel and Lin, 2002; Pantel et al., 2009). For instance, in the syntactic tree of sentence "*this is an interesting read for anyone studying logic*", one context of the word "logic" can be defined as "*study V:obj:N*". In this paper, we adopt Minipar (Lin, 1994) to parse sentences and to construct syntactic trees.

One popular *lexical context* is *text window*, where a context c for a term w in a sentence S is defined as a substring of the sentence containing but removing w. For example, for sentence "$\ldots w_1w_2w_3ww_4w_5w_6\ldots$", a text window context (with size 4) of w can be "$w_2w_3w_4w_5$". It is typically time-consuming to construct the syntactic trees for a large-scale dataset, even with a lightweight syntactic parser like Minipar. The construction of lexical contexts is much more efficient because it does not require the syntactic dependency between terms. Both contexts are studied in this paper.

After defining contexts for a term w, the next step is to construct a feature vector for the term: $F(w)=(f_{w1}, f_{w2}\ldots, f_{w,m})$, where m is the number of distinct contexts, and $f_{w,c}$ is the feature value of context c with respect to term w. Among all the existing approaches, the dominant way of assigning feature values (or context values) is computing the pointwise mutual information (PMI) between the feature and the term,

$$f_{w,c} = \mathrm{PMI}_{w,c} = \log\frac{F(w,c)\cdot F(*,*)}{F(w,*)\cdot F(*,c)} \qquad (3.1)$$

where $F(w,c)$ is the frequency of context c occurring for term w, $F(w,*)$ is the total frequency of all contexts for term w, $F(*,c)$ is the frequency of context c for all terms, and $F(*,*)$ is the total frequency of all context for all terms. They are calculated as follows respectively,

$$\begin{aligned} F(w,*) &= \textstyle\sum_{j=1}^{m} F(w,j) \\ F(*,c) &= \textstyle\sum_{i=1}^{n} F(i,c) \\ F(*,*) &= \textstyle\sum_{i=1}^{n}\sum_{j=1}^{m} F(i,j) \end{aligned} \qquad (3.2)$$

where m and n are respectively the distinct numbers of contexts and terms.

Following state-of-the-art, we adopt PMI in this paper for context weighting.

Given the feature vectors of terms, the similarity of any two terms is naturally computed as the similarity of their corresponding feature vectors. Cosine similarity and Jaccard similarity (weighted) are implemented in our experiments,

$$Cosine(\vec{x},\vec{y}) = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}\cdot\sqrt{\sum_i y_i^2}} \qquad (3.3)$$

$$Jaccard(\vec{x},\vec{y}) = \frac{\sum_i \min(x_i,y_i)}{\sum_i x_i + \sum_i y_i - \sum_i \min(x_i,y_i)} \quad (3.4)$$

Jaccard similarity is finally used in presenting our experimental results (in Section 6), because it achieves higher performance.

3.2 Pattern-based Approaches

In PB approaches, a list of carefully-designed (or automatically learned) patterns is exploited and applied to a text collection, with the hypothesis that the terms extracted by applying each of the patterns to a specific piece of text tend to be similar. Two categories of patterns have been studied in the literature: sentence lexical patterns, and HTML tag patterns. Table-2 lists some popular patterns utilized in existing semantic class mining work (Heast 1992; Pasca 2004; Kozareva et al., 2008; Zhang et al., 2009). In the table, "T" means a term (a word or a phrase). Exactly the same set of patterns is employed in implementing our pattern-based approaches in this paper.

Type	Pattern
Lexical	T {, T}* {,} (and|or) {other} T
	(such as | including) T (and|,|.)
	T, T, T {,T}*
Tag	- T … - T
	- T … - T
	<select> <option> T …<option> T </select>
	<table> <tr> <td> T </td> … <td> T </td> </tr> ... </table>
	Other Html-tag repeat patterns

Table 2. Patterns employed in this paper (Lexical: sentence lexical patterns; Tag: HTML tag patterns)

We call the set of terms extracted by applying a pattern one time as a *raw semantic class* (RASC). The term similarity graph needs to be built by aggregating the information of the extracted RASCs.

One basic idea of estimating term similarity is to count the number of RASCs containing both of them. This idea is extended in the state-of-the-art approaches (Zhang et al., 2009) to distinguish the reliability of different patterns and to punish term similarity contributions from the same domain (or site), as follows,

$$Sim(a,b) = \sum_{i=1}^{m} \log(1 + \sum_{j=1}^{k_i} w(P(C_{i,j}))) \quad (3.5)$$

where $C_{i,j}$ is a RASC containing both term a and term b, $P(C_{i,j})$ is the pattern via which the RASC is extracted, and $w(P)$ is the weight of pattern P. The above formula assumes all these RASCs belong to m sites (or domains) with $C_{i,j}$ extracted from a page in site i, and k_i being the number of RASCs corresponding to site i.

In this paper, we adopt an extension of the above formula which considers the frequency of a single term, as follows,

$$Sim^*(a,b) = Sim(a,b) \cdot \sqrt{IDF(a) \cdot IDF(b)} \quad (3.6)$$

where $IDF(a)=\log(1+N/N(a))$, N is the total number of RASCs, and $N(a)$ is the number of RASCs containing a. In the experiments, we simply set the weight of every pattern type to be the same value (1.0).

4 Compare PB and DS

We compare PB and DS by the quality of the term similarity graphs they generated. The quality of a term graph is measured by set expansion: Given a list of *seed* terms (e.g., S={*lent*, *epiphany*}) belonging to a semantic class, our task is to find other members of this class, such as *advent*, *easter*, and *christmas*.

In this section, we first describe our set expansion algorithm adopted in our study. Then DS and PB are compared in terms of their set-expansion performance. Finally we discuss ways of selecting appropriate approaches for different types of seeds to get better expansion results.

4.1 Set Expansion Algorithm

Having at hand the similarity graph, set expansion can be implemented by selecting the terms most similar to the seeds. So given a query Q={s_1, s_2, …, s_k}, the key is to compute $f(t,Q)$, the similarity between a term t and the seed-set Q. Naturally, we define it as the weighted average similarity between t and every seed in Q,

$$f(t,Q) = \sum_{i=1}^{k} w_i \cdot Sim(t,s_i) \quad (4.1)$$

where w_i is the weight of seed s_i, which can be a constant value, or a function of the frequency of term s_i in the corpus. Although Formula 3.6 can be adopted directly for calculating $Sim(t,s_i)$, we use the following rank-based formula because it generate better expansion results.

$$Sim(t,s_i) = \frac{1}{\log(\lambda + r(t,s_i))} \quad (4.2)$$

where $r(t,s_i)$ is the rank of term t among the neighbors of s_i.

In our experiments, we fix w_i=1 and λ=10.

4.2 Compare DS with PB

In order to have a comprehensive comparison of the two approaches, we intentionally choose terms of diverse types and do experiments based on various data scales. We classify terms into 5 types by their lexical categories: proper nouns, common nouns, verbs, adjectives, and adverbs. The data scales for experiments are from one million to 500 million web pages. Please refer to sections 5.1 and 5.2 for more details about the corpora and seeds used for experiments.

Experimental results (refer to Section 6) will show that, for proper nouns, the ranking of approaches (in terms of performance) is:

$$PB > PB\text{-}HtmlTag > DS \approx PB\text{-}Lexical$$

While for common nouns, verbs, adjectives, and adverbs, we have:

$$DS > PB$$

Here "PB-lexical" means only the lexical patterns of Table 2 are adopted. Similarly, "PB-HtmlTag" represents the PB approach with only Html-tag patterns being utilized.

Please pay attention that this paper by no means covers all PB or DS approaches (although we have tried our best to include the most popular ones). For PB, there are of course other kinds of patterns (e.g., patterns based on deeper linguistic analysis). For DS, other types of contexts may exist in addition to those listed in Table 1. So in interpreting experimental results, making observations, and drawing preliminary conclusions, we only means the patterns in Table 2 for PB and Table 1 for DS. It will be an interesting future work to include more DS and PB approaches in the study.

In order to understand why PB performs so well in dealing with proper nouns while so badly for other term categories, we calculated the frequency of each seed term in the extracted *RASCs*, the output of the pattern-matching algorithm. We define the *normalized frequency* of a term to be its frequency in the RASCs divided by the frequency in the sentences of the original documents (with duplicate sentences merged). Then we define the *mean normalized frequency* (MNF) of a seed set *S*, as follows,

$$MNF(S) = \frac{\sum_{t \in S} F_{norm}(t)}{|S|} \qquad (4.3)$$

where $F_{norm}(t)$ is the normalized frequency of t.

The MNF values for the five seed sets are listed in Table 3, where we can see that proper nouns have the largest MNF values, followed by common nouns. In other words, the patterns in Table 2 capture the relations of more proper nouns than other term categories.

Seed Categories	Terms	MNF
Proper nouns	40	0.2333
Common nouns	40	0.0716
Verbs	40	0.0099
Adjectives	40	0.0126
Adverbs	40	0.0053

Table 3. MNF values of different seed categories

As mentioned in the introduction, the PB and DS approaches we studied capture first-order and second-order term co-occurrences respectively. Some existing work (e.g., Edmonds, 1997) showed that second-order co-occurrence leads to better results for detecting synonymy. Considering that a high proportion of coordinate terms of verbs, adjectives, and adverbs are their synonyms and antonyms, it is reasonable that DS behaves better for these term types because it exploits second-order co-occurrence. For PB, different from the standard way of dealing with first-order co-occurrences where statistics are performed on all pairs of near terms, a *subset* of co-occurred terms are *selected* in PB by specific patterns. The patterns in Table-2 help detecting coordinate proper nouns, because they are frequently occurred together obeying the patterns in sentences or web pages. But it is not the case for other term types. It will be interesting to study the performance of PB when more pattern types are added.

4.3 Approach Selection

Having observed that the two approaches perform quite differently on every type of queries we investigated, we hope we can improve the expansion performance by smartly selecting an approach for each query. In this section, we propose and study several approach-selection methods, by which we hope to gain some insights about the possibility and effectiveness of combining DS and PB for better set expansion.

Oracle selection: In order to get an insight about the upper bound that we could obtain when combing the two methods, we implement an oracle that chooses, for each query, the approach that generates better expansion results.

Frequency-based selection: It is shown in Table 3 that the *mean normalized frequency* of proper nouns is much larger than other terms. Motivated by this observation, we select a set expansion methodology for each query as follows: Select PB if the normalized frequency values of all terms in the query are larger than 0.1; otherwise choose DS.

We demonstrate, in Section 6.3, the effectiveness of the above selection methods.

5 Experimental Setup

5.1 Dataset and Exp. Environment

We adopt a public-available dataset in our experiments: ClueWeb09[4]. This is a very large dataset collected by Carnegie Mellon University in early 2009 and has been used by several tracks of the Text Retrieval Conference (TREC)[5]. The whole dataset consists of 1.04 billion web pages in ten languages while only those in English, about 500 million pages, are used in our experiments. The reason for selecting such a dataset is twofold: First, it is a corpus large enough for conducting web-scale experiments and getting meaningful results. Second, since it is publicly available, it is possible for other researchers to reproduce the experiments in the paper.

Corpora	Docs (millions)	Sentences (millions)	Description
Clue500	500	13,000	All En pages in ClueWeb09
Clue050	50	1,600	ClueWeb09 category B
Clue010	10	330	Sampling from Clue050
Clue001	1	42	Sampling from Clue050

Table 4. Corpora used in experiments

To test the impact of corpus size on set expansion performance, four corpora are derived from the dataset, as outlined in Table 4. The Clue500 corpus contains all the 500 million English web pages in the dataset; while Clue050 is a subset of ClueWeb09 (named category B) containing 50 million English web pages. The remaining two corpora are respectively the 1/5 and 1/50 random sampling of web pages from Clue050.

Documents in the corpora are stored and processed in a cluster of 40 four-core machines.

[4] http://boston.lti.cs.cmu.edu/Data/clueweb09/
[5] http://trec.nist.gov/

5.2 Query Sets

We perform our study using two query sets.

WikiGold: It was collected by Pantel et al. (2009) from the "List of" pages in Wikipedia and used as the gold standard in their paper. This gold standard consists of 49 entity sets, and 20220 trials (used as queries) of various numbers of seeds. Most seeds in the query set are named entities. Please refer to Pantel et al. (2009) for details of the gold standard.

Mix100: This query set consists of 100 queries in five categories: verbs, adjectives, adverbs, common nouns, and proper nouns. There are 20 queries in every category and two seeds in every query. The query set was built by the following steps: First, 20 terms of each category were randomly selected from a term list (which is constructed by part-of-speech tagging the Clue050 corpus and removing low-frequency terms), and were treated as the first seed of the each query. Then, we manually added one additional seed for each query. The reason for utilizing two seeds instead of one is the observation that a large portion of the terms selected in the previous step belong to multiple categories. For example, "*colorful*" is both an adjective and a proper noun (a Japanese manga).

5.3 Results Labeling

No human labeling efforts are needed for the expansion results of the WikiGold query set. Every returned term is automatically judged to be "Good" (otherwise "Bad") if it appears in the corresponding gold standard entity set.

For Mix100, the search results of various approaches are merged and labeled by three human labelers. Each labeler assigns each term in the search results a label of "Good", "Fair" or "Bad". The labeling agreement values (measured by percentage agreement) between labelers I and II, I and III, II and III are respectively 0.82, 0.81, and 0.81. The ultimate judgment of each result term is obtained from the three labelers by majority voting. In the case of three labelers giving mutually different results (i.e., one "Good", one "Fair" and one "Bad"), the ultimate judgment is set to "Fair" (the average).

5.4 Evaluation Metrics

After removing seeds from the expansion results, we adopt the following metrics to evaluate the

results of each query. The evaluation score on a query set is the average over all the queries.

Precision@*k*: The percentage of relevant (good or fair) terms in the top-*k* expansion results (terms labeled as "Fair" are counted as 0.5)

Recall@*k*: The ratio of relevant terms in the top-*k* results to the total number of relevant terms

R-Precision: Precision@R where *R* is the total number of terms labeled as "Good"

Mean average precision (MAP): The average of precision values at the positions of all good or fair results

6 Experimental Results

6.1 Overall Performance Comparison

Table 5 lists the performance (measured by MAP, R-precision, and the precisions at ranks 25, 50, and 100) of some key approaches on corpus Clue050 and query set WikiGold. The results of query set Mix100 are shown in Table 6. In the results, *TWn* represents the DS approach with text-window of size *n* as contexts, *Syntactic* is the DS approach with syntactic contexts, *PB-Lexical* means only the lexical patterns of Table 2 are adopted, and *PB-HtmlTag* represents the PB approach with only Html-tag patterns utilized.

Approach	MAP	R-Prec	P@25	P@50	P@100
TW2	0.218	0.287	0.359	0.278	0.204
TW4	0.152	0.210	0.325	0.244	0.173
Syntactic	0.170	0.247	0.314	0.242	0.178
PB-Lexical	0.227	0.276	0.352	0.272	0.190
PB-HtmlTag	0.354	0.417	0.513	0.413	0.311
PB	**0.362**	**0.424**	**0.520**	**0.418**	**0.314**
Pantel-24M	N/A	0.264	0.353	0.298	0.239
Pantel-120M	N/A	0.356	0.377	0.319	0.250
Pantel-600M	N/A	0.404	0.407	0.347	0.278

Table 5. Performance comparison on the Clue050 corpus (query set: WikiGold)

It is shown that PB gets much higher evaluation scores than other approaches on the *WikiGold* query set and the proper-nouns category of *Mix100*. While for other seed categories in *Mix100*, TW2 return significantly better results. We noticed that most seeds in *WikiGold* are proper nouns. So the experimental results tend to indicate that the performance comparison between state-of-the-art DS and PB approaches depends on the types of terms to be mined, specifically, DS approaches perform better in mining semantic classes of common nouns, verbs, adjectives, and adverbs; while state-of-the-art PB approaches are more suitable for mining semantic classes of proper nouns. The performance of PB is low in dealing with other types of terms (especially adverbs). The performance of PB drops significantly if only lexical patterns are used; and the HtmlTag-only version of PB performs only slightly worse than PB.

The observations are verified by the precision-recall graph in Figure 1 on Clue500. The results of the *syntactic* approach on Clue500 are not included, because it is too time-consuming to parse all the 500 million web pages by a dependency parser (even using a high-performance parser like Minipar). It took overall about 12,000 CPU-hours to parse all the sentences in Clue050 by Minipar.

Query types & Approaches		MAP	P@5	P@10	P@20
Proper Nouns	TW2	0.302	0.835	0.810	0.758
	PB	**0.336**	**0.920**	**0.838**	**0.813**
Common Nouns	TW2	**0.384**	**0.735**	**0.668**	**0.595**
	PB	0.212	0.640	0.548	0.485
Verbs	TW2	**0.273**	**0.655**	**0.543**	**0.465**
	PB	0.176	0.415	0.373	0.305
Adjectives	TW2	**0.350**	**0.655**	**0.563**	**0.473**
	PB	0.120	0.335	0.285	0.234
Adverbs	TW2	**0.432**	**0.605**	**0.505**	**0.454**
	PB	0.043	0.100	0.095	0.089

Table 6. Performance comparison on different query types (Corpus: Clue050; query set: Mix100)

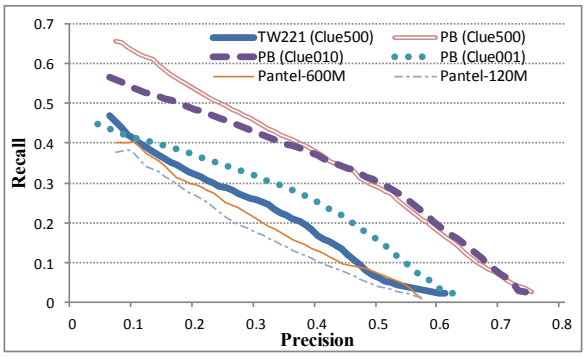


Figure 1. Precision and recall of various approaches (query set: WikiGold)

The methods labeled Pantel-24M etc. (in Table 5 and Figure 1) are the approaches presented in (Pantel et al., 2009) on their corpus (called Web04, Web20, and Web100 in the paper) containing respectively 24 million, 120 million, and 600 million web pages. Please pay attention that their results and ours may not be directly comparable, because different corpora and set-

expansion algorithms were used. Their results are listed here for reference purpose only.

6.2 Corpus Size Effect

Table 7 shows the performance (measured by MAP) of two approaches on query set *Mix100*, by varying corpus size. We observed that the performance of TW2 improves rapidly along with the growth of corpus size from one million to 50 million documents. From Clue050 to Clue500, the performance is slightly improved.

Query types & Approaches		Clue001	Clue010	Clue050	Clue500
Proper Nouns	TW2	0.209	0.265	0.302	**0.311**
	PB	**0.355**	0.351	0.336	0.327
Common Nouns	TW2	0.259	0.348	0.384	**0.393**
	PB	0.200	**0.234**	0.212	0.205
Verbs	TW2	0.224	0.268	0.273	**0.278**
	PB	0.101	0.134	**0.176**	0.148
Adjectives	TW2	0.309	0.326	0.350	**0.353**
	PB	0.077	**0.158**	0.120	0.129
Adverbs	TW2	0.413	0.423	0.432	**0.437**
	PB	0.028	0.058	0.043	**0.059**

Table 7. Effect of different corpus size (query set: Mix100; metric: MAP)

For PB, however, the performance change is not that simple. For proper nouns, the best performance (in terms of MAP) is got on the two small corpora Clue001 and Clue010; and the score does not increase when corpus size grows. Different observations are made on WikiGold (see Figure 1), where the performance improves a lot with the data growth from Clue001 to Clue010, and then stabilizes (from Clue010 to Clue500). For other term types, the MAP scores do not grow much after Clue010. To our current understanding, the reason may be due to the two-fold effect of incorporating more data in mining: *bringing useful information as well as noise*. Clue001 contains enough information, which is fully exploited by the PB approach, for expanding the proper-nouns in Mix100. So the performance of PB on Clue001 is excellent. The named entities in WikiGold are relatively rare, which requires a larger corpus (Clue010) for extracting peer terms from. But when the corpus gets larger, we may not be able to get more useful information to further improve results quality.

Another interesting observation is that, for proper nouns, the performance of PB on Clue001 is even much better than that of TW2 on corpus Clue500. Similarly, for other query types (common nous, verbs, adjectives, and adverbs), TW2 easily beats PB even with a much smaller corpus.

6.3 Approach Selection

Here we demonstrate the experimental results of combining DS and PB with the methods we proposed in Section 4.3. Table 8 shows the combination of PB and TW2 on corpus Clue050 and query set Mix100. The overall performance relies on the number (or percentage) of queries in each category. Two ways of mixing the queries are tested: avg(4:1:1:1:1) and avg(1:1:1:1:1), where the numbers are the proportion of proper nouns, common nouns, verbs, adjectives, and adverbs.

Approach	Avg (1:1:1:1:1)			Avg (4:1:1:1:1)		
	P@5	P@10	P@20	P@5	P@10	P@20
TW2	0.697	0.618	0.548	0.749	0.690	0.627
PB	0.482	0.428	0.385	0.646	0.581	0.545
Oracle	0.759	0.663	0.591	0.836	0.759	0.695
Freq-based	0.721	0.633	0.570	0.799	0.723	0.671

Table 8. Experiments of combining both approaches (Corpus: Clue050; query set: Mix100)

The expansion performance is improved a lot with our frequency-based combination method. As expected, oracle selection achieves great performance improvement, which shows the large potential of combining DS and PB. Similar results (omitted due to space limitations) are observed on the other corpora.

Our online semantic mining system (*NeedleSeek*, http://needleseek.msra.cn) adopts both PB and DS for semantic class construction.

7 Conclusion

We compared two mainstream methods (DS and PB) for semantic class mining, based on a dataset of 500 million pages and using five term types. We showed that PB is clearly adept at extracting semantic classes of proper nouns; while DS is relatively good at dealing with other types of terms. In addition, a small corpus is sufficient for each approach to generate better semantic classes of its "favorite" term types than those obtained by its counterpart on a much larger corpus. Finally, we tried a frequency-based method of combining them and saw apparent performance improvement.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Pasca, Aitor Soroa. A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches. NAACL-HLT 2009.

Philip Edmonds. 1997. Choosing the Word most Typical in Context Using a Lexical Co-Occurrence Network. *ACL'97*, pages 507-509.

Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel Weld, and Alexander Yates. 2004. Web-Scale Information Extraction in KnowItAll. *WWW'04*, pages 100–110, New York.

Zelig S. Harris. 1985. Distributional Structure. *The Philosophy of Linguistics*. New York: Oxford University Press.

Marti A. Hearst. 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. *COLING'92*, Nantes, France.

Donald Hindle. 1990. Noun Classification from Predicate-Argument Structures. In ACL'90, pages 268–275, Pittsburg, Pennsylvania, June.

Mamoru Komachi and Hisami Suzuki. Minimally Supervised Learning of Semantic Knowledge from Query Logs. *IJCNLP 2008*, pages 358–365, 2008.

Zornitsa Kozareva, Ellen Riloff, Eduard Hovy. 2008. Semantic Class Learning from the Web with Hyponym Pattern Linkage Graphs. ACL'08: HLT.

Dekang Lin. 1994. Principar - an Efficient, Broad-Coverage, Principle-based Parser. *COLING'94*, pp. 482-488.

Dekang Lin. 1998. Automatic Retrieval and Clustering of Similar Words. *COLING-ACL'98*, pages 768-774.

Dekang Lin and Patrick Pantel. 2001. Induction of Semantic Classes from Natural Language Text. *SIGKDD'01*, pages 317-322.

Hiroaki Ohshima, Satoshi Oyama and Katsumi Tanaka. 2006. Searching Coordinate Terms with their Context from the Web. *WISE'06*.

Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu and Vishnu Vyas. 2009. Web-Scale Distributional Similarity and Entity Set Expansion. *EMNLP'09*. Singapore.

Patrick Pantel and Dekang Lin. 2002. Discovering Word Senses from Text. *SIGKDD'02*.

Patric Pantel, Deepak Ravichandran, and Eduard Hovy. 2004. Towards Terascale Knowledge Acquisition. *COLING'04*, Geneva, Switzerland.

Marius Pasca. 2004. Acquisition of Categorized Named Entities for Web Search. *CIKM'04*.

Marius Pasca. 2007. Weakly-Supervised Discovery of Named Entities Using Web Search Queries. *CIKM'07*. pp. 683-690.

Marius Pasca and Benjamin Van Durme. 2008. Weakly-supervised Acquisition of Open-Domain Classes and Class Attributes from Web Documents and Query Logs. *ACL'08*.

Marius Pasca, Dekang Lin, Jeffrey Bigham, Andrei Lifchits, and Alpa Jain. 2006. Organizing and Searching the World Wide Web of Facts - Step One: The One-Million Fact Extraction Challenge. In AAAI'06.

Marco Pennacchiotti and Patrick Pantel. 2009. Entity Extraction via Ensemble Semantics. EMNLP'09.

Gerda Ruge. 1992. Experiments on Linguistically-Based Term Associations. *Information Processing & Management*, 28(3): 317-32.

Keiji Shinzato and Kentaro Torisawa. 2005. A Simple WWW-based Method for Semantic Word Class Acquisition. *Recent Advances in Natural Language Processing (RANLP'05)*, Borovets, Bulgaria.

Shuming Shi, Xiaokang Liu, Ji-Rong Wen. 2008. Pattern-based Semantic Class Discovery with Multi-Membership Support. *CIKM'08*, Napa Valley, California, USA.

Hristo Tanev and Bernardo Magnini. 2006. Weakly Supervised Approaches for Ontology Population. EACL'2006, Trento, Italy.

Richard C. Wang and William W. Cohen. 2008. Iterative Set Expansion of Named Entities Using the Web. *ICDM'08*, pages 1091–1096.

Masashi Yamaguchi, Hiroaki Ohshima, Satoshi Oyama, and Katsumi Tanaka. Unsupervised Discovery of Coordinate Terms for Multiple Aspects from Search Engine Query Logs. *The 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*.

Huibin Zhang, Mingjie Zhu, Shuming Shi, and Ji-Rong Wen. 2009. Employing Topic Models for Pattern-based Semantic Class Discovery. *ACL'09*, Singapore.

Metaphor Identification Using Verb and Noun Clustering

Ekaterina Shutova, Lin Sun and Anna Korhonen
Computer Laboratory, University of Cambridge
es407,ls418,alk23@cam.ac.uk

Abstract

We present a novel approach to automatic metaphor identification in unrestricted text. Starting from a small seed set of manually annotated metaphorical expressions, the system is capable of harvesting a large number of metaphors of similar syntactic structure from a corpus. Our method is distinguished from previous work in that it does not employ any hand-crafted knowledge, other than the initial seed set, but, in contrast, captures metaphoricity by means of verb and noun clustering. Being the first to employ unsupervised methods for metaphor identification, our system operates with the precision of 0.79.

1 Introduction

Besides enriching our thought and communication with novel imagery, the phenomenon of metaphor also plays a crucial structural role in our use of language. Metaphors arise when one concept is viewed in terms of the properties of the other. Below are some examples of metaphor.

(1) How can I *kill* a process? (Martin, 1988)

(2) Inflation has *eaten up* all my savings. (Lakoff and Johnson, 1980)

(3) He *shot down* all of my arguments. (Lakoff and Johnson, 1980)

(4) And then my heart with pleasure *fills*,
And *dances* with the daffodils.[1]

In metaphorical expressions seemingly unrelated features of one concept are associated with another concept. In the computer science metaphor in (1) the *computational process* is viewed as something *alive* and, therefore, its forced termination is associated with the act of killing. Lakoff and Johnson (1980) explain metaphor as a systematic association, or a *mapping*, between two concepts or conceptual domains: the *source* and the *target*. The metaphor in (3) exemplifies a mapping of a concept of *argument* to that of *war*. The *argument*, which is the target concept, is viewed in terms of a *battle* (or a *war*), the source concept. The existence of such a link allows us to talk about arguments using the war terminology, thus giving rise to a number of metaphors.

Characteristic to all areas of human activity (from poetic to ordinary to scientific) and, thus, to all types of discourse, metaphor becomes an important problem for natural language processing (NLP). In order to estimate the frequency of the phenomenon, Shutova and Teufel (2010) conducted a corpus study on a subset of the British National Corpus (BNC) (Burnard, 2007) representing various genres. They manually annotated metaphorical expressions in this data and found that 241 out of 761 sentences contained a metaphor, whereby in 164 phrases metaphoricity was introduced by a verb. Due to such a high frequency of their use, a system capable of recognizing and interpreting metaphorical expressions in unrestricted text would become an invaluable component of any semantics-oriented NLP application.

Automatic processing of metaphor can be clearly divided into two subtasks: *metaphor identification* (distinguishing between literal and metaphorical language in text) and *metaphor interpretation* (identifying the intended literal meaning of a metaphorical expression). Both of them have been repeatedly attempted in NLP.

To date the most influential account of metaphor identification is that of Wilks (1978).

[1] "I wandered lonely as a cloud", William Wordsworth, 1804.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1002–1010,
Beijing, August 2010

According to Wilks, metaphors represent a violation of selectional restrictions in a given context. Consider the following example.

(5) My car *drinks* gasoline. (Wilks, 1978)

The verb *drink* normally takes an *animate* subject and a *liquid* object. Therefore, *drink* taking a *car* as a subject is an anomaly, which may as well indicate metaphorical use of *drink*.

This approach was automated by Fass (1991) in his met* system. However, Fass himself indicated a problem with the method: it detects any kind of non-literalness or anomaly in language (metaphors, metonymies and others), i.e., it overgenerates with respect to metaphor. The techniques met* uses to differentiate between those are mainly based on hand-coded knowledge, which implies a number of limitations. In a similar manner manually created knowledge in the form of WordNet (Fellbaum, 1998) is employed by the system of Krishnakumaran and Zhu (2007), which essentially differentiates between highly lexicalized metaphors included in WordNet, and novel metaphorical senses.

Alternative approaches (Gedigan et al., 2006) search for metaphors of a specific domain defined a priori (e.g. MOTION metaphors) in a specific type of discourse (e.g. Wall Street Journal). In contrast, the scope of our experiments is the whole of the British National Corpus (BNC) (Burnard, 2007) and the domain of the expressions we identify is unrestricted. However, our technique is also distinguished from the systems of Fass (1991) and Krishnakumaran and Zhu (2007) in that it does not rely on any hand-crafted knowledge, but rather captures metaphoricity in an unsupervised way by means of verb and noun clustering.

The motivation behind the use of clustering methods for metaphor identification task lies in the nature of metaphorical reasoning based on association. Compare, for example, the target concepts of *marriage* and *political regime*. Having quite distinct meanings, both of them are cognitively mapped to the source domain of *mechanism*, which shows itself in the following examples:

(6) Our relationship is not really *working*.

(7) Diana and Charles did not succeed in *mending* their marriage.

(8) The *wheels* of Stalin's regime were *well oiled* and already *turning*.

We expect that such relatedness of distinct target concepts should manifest itself in the examples of language use, i.e. target concepts that are associated with the same source concept should appear in similar lexico-syntactic environments. Thus, clustering concepts using grammatical relations (GRs) and lexical features would allow us to capture their relatedness **by association** and harvest a large number of metaphorical expressions beyond our seed set. For example, the sentence in (6) being part of the seed set should enable the system to identify metaphors in both (7) and (8).

In summary, our system (1) starts from a seed set of metaphorical expressions exemplifying a range of source–target domain mappings; (2) performs unsupervised noun clustering in order to harvest various target concepts associated with the same source domain; (3) by means of unsupervised verb clustering creates a source domain verb lexicon; (4) searches the BNC for metaphorical expressions describing the target domain concepts using the verbs from the source domain lexicon.

We tested our system starting with a collection of metaphorical expressions representing verb-subject and verb-object constructions, where the verb is used metaphorically. We evaluated the precision of metaphor identification with the help of human judges. In addition to this we compared our system to a baseline built upon WordNet, whereby we demonstrated that our method goes far beyond synonymy and captures metaphors not directly related to any of those seen in the seed set.

2 Experimental Data

2.1 Seed Phrases

We used the dataset of Shutova (2010) as a seed set. Shutova (2010) annotated metaphorical expressions in a subset of the BNC sampling various genres: literature, newspaper/journal articles, essays on politics, international relations and history, radio broadcast (transcribed speech). The dataset consists of 62 phrases that are single-word

metaphors representing verb-subject and verb-object relations, where a verb is used metaphorically. The seed phrases include e.g. *stir excitement, reflect enthusiasm, accelerate change, grasp theory, cast doubt, suppress memory, throw remark* (verb - direct object constructions) and *campaign surged, factor shaped [..], tension mounted, ideology embraces, changes operated, approach focuses, example illustrates* (subject - verb constructions).

2.2 Corpus

The search space for metaphor identification was the British National Corpus (BNC) that was parsed using the RASP parser of Briscoe et al. (2006). We used the grammatical relations output of RASP for BNC created by Andersen et al. (2008). The system searched the corpus for the source and target domain vocabulary within a particular grammatical relation (verb-object or verb-subject).

3 Method

Starting from a small seed set of metaphorical expressions, the system implicitly captures the associations that underly their production and comprehension. It generalizes over these associations by means of unsupervised verb and noun clustering. The obtained clusters then represent potential source and target concepts between which metaphorical associations hold. The knowledge of such associations is then used to annotate metaphoricity in a large corpus.

3.1 Clustering Motivation

Abstract concepts that are associated with the same source domain are often related to each other on an intuitive and rather structural level, but their meanings, however, are not necessarily synonymous or even semantically close. The results of previous research on corpus-based lexical semantics suggest that the linguistic environment in which a lexical item occurs can shed light on its meaning. A number of works have shown that it is possible to automatically induce semantic word classes from corpus data via clustering of contextual cues (Pereira et al., 1993; Lin, 1998; Schulte im Walde, 2006). The consensus is that the lexical items exposing similar behavior in a large body of text most likely have the same meaning. However, the concepts of *marriage* and *political regime*, that are also observed in similar lexico-syntactic environments, albeit having quite distinct meanings are likewise assigned by such methods to the same cluster. In contrast to concrete concepts, such as *tea, water, coffee, beer, drink, liquid*, that are clustered together due to **meaning similarity**, abstract concepts tend to be clustered together **by association** with the same source domain. It is the presence of this association that explains the fact that they share common contexts. We exploit this idea for identification of new target domains associated with the same source domain. We then use unsupervised verb clustering to collect source domain vocabulary, which in turn allows us to harvest a large number of new metaphorical expressions.

3.2 Verb and Noun Clustering

Since Levin (1993) published her classification, there have been a number of attempts to automatically classify verbs into semantic classes using supervised and unsupervised approaches (Lin, 1998; Brew and Schulte im Walde, 2002; Korhonen et al., 2003; Schulte im Walde, 2006; Joanis et al., 2008; Sun and Korhonen, 2009). Similar methods were also applied to acquisition of noun classes from corpus data (Rooth et al., 1999; Pantel and Lin, 2002; Bergsma et al., 2008).

We adopt a recent verb clustering approach of Sun and Korhonen (2009), who used rich syntactic and semantic features extracted using a shallow parser and a clustering method suitable for the resulting high dimensional feature space. When Sun and Korhonen evaluated their approach on 204 verbs from 17 Levin classes, they obtained 80.4 F-measure (which is high in particular for an unsupervised approach). We apply this approach to a much larger set of 1610 verbs: all the verb forms appearing in VerbNet (Kipper et al., 2006) with the exception of highly infrequent ones. In addition, we adapt the approach to noun clustering.

3.2.1 Feature Extraction

Our verb dataset is a subset of VerbNet compiled as follows. For all the verbs in VerbNet we

extracted their occurrences (up to 10,000) from the raw corpus data collected originally by Korhonen et al. (2006) for construction of VALEX lexicon. Only the verbs found in this data more than 150 times were included in the experiment.

For verb clustering, we adopted the best performing features of Sun and Korhonen (2009): automatically acquired verb subcategorization frames (SCFs) parameterized by their selectional preferences (SPs). We obtained these features using the SCF acquisition system of Preiss et al. (2007). The system tags and parses corpus data using the RASP parser and extracts SCFs from the resulting GRs using a rule-based classifier which identifies 168 SCF types for English verbs. It produces a lexical entry for each verb and SCF combination occurring in corpus data. We obtained SPs by clustering argument heads appearing in the subject and object slots of verbs in the resulting lexicon.

Our noun dataset consists of 2000 most frequent nouns in the BNC. Following previous works on semantic noun classification (Pantel and Lin, 2002; Bergsma et al., 2008), we used GRs as features for noun clustering. We employed all the argument heads and verb lemmas appearing in the subject, direct object and indirect object relations in the RASP-parsed BNC.

The feature vectors were first constructed from the corpus counts, and subsequently normalized by the sum of the feature values before applying clustering.

3.2.2 Clustering Algorithm

We use spectral clustering (SPEC) for both verbs and nouns. This technique has proved to be effective in previous verb clustering works (Brew and Schulte im Walde, 2002; Sun and Korhonen, 2009) and in related NLP tasks involving high dimensional data (Chen et al., 2006). We use the MNCut algorithm for SPEC which has a wide applicability and a clear probabilistic interpretation (Meila and Shi, 2001).

The task is to group a given set of words $W = \{w_n\}_{n=1}^{N}$ into a disjoint partition of K classes. SPEC takes a similarity matrix as input. We construct it using the Jensen-Shannon divergence (JSD) as a measure. The JSD between two feature vectors w and w' is $d_{jsd}(w, w') = \frac{1}{2}D(w||m) + \frac{1}{2}D(w'||m)$ where D is the Kullback-Leibler divergence, and m is the average of the w and w'.

The similarity matrix S is constructed where $S_{ij} = \exp(-d_{jsd}(w, w'))$. In SPEC, the similarities S_{ij} are viewed as weights on the edges ij of a graph G over W. The similarity matrix S is thus the adjacency matrix for G. The degree of a vertex i is $d_i = \sum_{j=1}^{N} S_{ij}$. A cut between two partitions A and A' is defined to be $\text{Cut}(A, A') = \sum_{m \in A, n \in A'} S_{mn}$.

The similarity matrix S is then transformed into a stochastic matrix P.

$$P = D^{-1}S \quad (1)$$

The degree matrix D is a diagonal matrix where $D_{ii} = d_i$.

It was shown by Meila and Shi (2001) that if P has the K leading eigenvectors that are piecewise constants[2] with respect to a partition I^* and their eigenvalues are not zero, then I^* minimizes the multiway normalized cut (MNCut):

$$\text{MNCut}(I) = K - \sum_{k=1}^{K} \frac{\text{Cut}(I_k, I_k)}{\text{Cut}(I_k, I)}$$

P_{mn} can be interpreted as the transition probability between the vertexes m, n. The criterion can thus be expressed as $\text{MNCut}(I) = \sum_{k=1}^{K}(1 - P(I_k \rightarrow I_k|I_k))$ (Meila, 2001), which is the sum of transition probabilities across different clusters. This criterion finds the partition where random walks are most likely to happen within the same cluster. In practice, the leading eigenvectors of P are not piecewise constants. However, we can extract the partition by finding the approximately equal elements in the eigenvectors using a clustering algorithm, such as K-Means.

Since SPEC has elements of randomness, we ran the algorithm multiple times and the partition that minimizes the distortion (the distances to cluster centroid) is reported. Some of the clusters obtained as a result of applying the algorithm to our noun and verb datasets are demonstrated in Figures 1 and 2 respectively. The noun clusters represent target concepts that we expect to be associated with the same source concept (some suggested source concepts are given in Figure 1, although the system only captures those implicitly).

[2] An eigenvector v is piecewise constant with respect to I if $v(i) = v(j) \forall i, j \in I_k$ and $k \in 1, 2...K$

Source: MECHANISM
Target Cluster: consensus relation tradition partnership resistance foundation alliance friendship contact reserve unity link peace bond myth identity hierarchy relationship connection balance marriage democracy defense faith empire distinction coalition regime division
Source: STORY; JOURNEY
Target Cluster: politics practice trading reading occupation profession sport pursuit affair career thinking life
Source: LOCATION; CONTAINER
Target Cluster: lifetime quarter period century succession stage generation decade phase interval future
Source: LIVING BEING; END
Target Cluster: defeat fall death tragedy loss collapse decline disaster destruction fate

Figure 1: Clustered target concepts

Source Cluster: sparkle glow widen flash flare gleam darken narrow flicker shine blaze bulge
Source Cluster: gulp drain stir empty pour sip spill swallow drink pollute seep flow drip purify ooze pump bubble splash ripple simmer boil tread
Source Cluster: polish clean scrape scrub soak
Source Cluster: kick hurl push fling throw pull drag haul
Source Cluster: rise fall shrink drop double fluctuate dwindle decline plunge decrease soar tumble surge spiral boom

Figure 2: Clustered verbs (source domains)

The verb clusters contain coherent lists of source domain vocabulary.

3.3 Selectional Preference Strength Filter

Following Wilks (1978), we take metaphor to represent a violation of selectional restrictions. However, not all verbs have an equally strong capacity to constrain their arguments, e.g. *remember, accept, choose* etc. are weak in that respect. We suggest that for this reason not all the verbs would be equally prone to metaphoricity, but only the ones exhibiting strong selectional preferences. We test this hypothesis experimentally and expect that placing this criterion would enable us to filter out a number of candidate expressions, that are less likely to be used metaphorically.

We automatically acquired selectional preference distributions for `Verb-Subject` and `Verb-Object` relations from the BNC parsed by RASP. We first clustered 2000 most frequent nouns in the BNC into 200 clusters using SPEC as described in the previous section. The obtained clusters formed our selectional preference classes. We adopted the selectional preference measure proposed by Resnik (1993) and successfully applied to a number of tasks in NLP including word sense disambiguation (Resnik, 1997). Resnik models selectional preference of a verb in probabilistic terms as the difference between the posterior distribution of noun classes in a particular relation with the verb and their prior distribution in that syntactic position regardless of the identity of the predicate. He quantifies this difference using the relative entropy (or Kullback-Leibler distance), defining the *selectional preference strength* (SPS) as follows.

$$S_R(v) = D(P(c|v)||P(c)) = \sum_c P(c|v) \log \frac{P(c|v)}{P(c)}, \quad (2)$$

where $P(c)$ is the prior probability of the noun class, $P(c|v)$ is the posterior probability of the noun class given the verb and R is the grammatical relation in question. SPS measures how strongly the predicate constrains its arguments.

We use this measure to filter out the verbs with weak selectional preferences. The optimal SPS threshold was set experimentally on a small held-out dataset and approximates to 1.32. We excluded expressions containing the verbs with preference strength below this threshold from the set of candidate metaphors.

4 Evaluation and Discussion

In order to prove that our metaphor identification method generalizes well over the seed set and goes far beyond synonymy, we compared its output to that of a baseline taking WordNet synsets to represent source and target domains. We evaluated the quality of metaphor tagging in terms of precision with the help of human judges.

4.1 Comparison against WordNet Baseline

The baseline system was implemented using synonymy information from WordNet to expand on the seed set. Assuming all the synonyms of the verbs and nouns in seed expressions to represent the source and target vocabularies respectively, the system searches for phrases composed of lexical items belonging to those vocabularies. For example, given a seed expression *stir excitement*, the baseline finds phrases such as *arouse fervour,*

stimulate agitation, stir turmoil etc. However, it is not able to generalize over the concepts to broad semantic classes, e.g. it does not find other *feelings* such as *rage, fear, anger, pleasure* etc., which is necessary to fully characterize the target domain. The same deficiency of the baseline system manifests itself in the source domain vocabulary: the system has only the knowledge of direct synonyms of *stir*, as opposed to other verbs characteristic to the domain of *liquids*, e.g. *pour, flow, boil* etc., successfully identified by means of clustering.

To compare the **coverage** achieved by unsupervised clustering to that of the baseline in quantitative terms, we estimated the number of WordNet synsets, i.d. different word senses, in the metaphorical expressions captured by the two systems. We found that the baseline system covers only 13% of the data identified using clustering and does not go beyond the concepts present in the seed set. In contrast, most metaphors tagged by our method are novel and represent a considerably wider range of meanings, e.g. given the seed metaphors *stir excitement, throw remark, cast doubt* the system identifies previously unseen expressions *swallow anger, hurl comment, spark enthusiasm* etc. as metaphorical.

4.2 Comparison with Human Judgements

In order to access the **quality** of metaphor identification by both systems we used the help of human annotators. The annotators were presented with a set of randomly sampled sentences containing metaphorical expressions as annotated by the system and by the baseline. They were asked to mark the tagged expressions that were metaphorical in their judgement as correct.

The annotators were encouraged to rely on their own intuition of metaphor. However, we also provided some guidance in the form of the following definition of metaphor[3]:

1. For each verb establish its meaning in context and try to imagine a more basic meaning of this verb on other contexts. Basic meanings normally are: (1) more concrete; (2) related to bodily action; (3) more precise (as opposed to vague); (4) historically older.

2. If you can establish the basic meaning that is distinct from the meaning of the verb in this context, the verb is likely to be used metaphorically.

[3]taken from the annotation procedure of Shutova and Teufel (2010) that is in turn partly based on the work of Pragglejaz Group (2007).

CKM 391 Time and time again he would stare at the ground, hand on hip, if he thought he had received a bad call, and then **swallow his anger** and play tennis.
AD9 3205 He tried to **disguise the anxiety** he felt when he found the comms system down, but Tammuz was nearly hysterical by this stage.
AMA 349 We will **halt the reduction** in NHS services for long-term care and community health services which support elderly and disabled patients at home.
ADK 634 **Catch their interest** and **spark their enthusiasm** so that they begin to see the product's potential.
K2W 1771 The committee heard today that gangs regularly **hurled** abusive **comments** at local people, making an unacceptable level of noise and leaving litter behind them.

Figure 3: Sentences tagged by the system (metaphors in bold)

We had 5 volunteer annotators who were all native speakers of English and had no or sparse linguistic knowledge. Their agreement on the task was 0.63 in terms of κ (Siegel and Castellan, 1988), whereby the main source of disagreement was the presence of highly lexicalized metaphors, e.g. verbs such as *adopt, convey, decline* etc. We then evaluated the system performance against their judgements in terms of *precision*. Precision measures the proportion of metaphorical expressions that were tagged correctly among the ones that were tagged. We considered the expressions tagged as metaphorical by at least three annotators to be correct. As a result our system identifies metaphor with the precision of 0.79, whereas the baseline only attains 0.44. Some examples of sentences annotated by the system are shown in Figure 3.

Such a striking discrepancy between the performance levels of the clustering approach and the baseline can be explained by the fact that a large number of metaphorical senses are included in WordNet. This means that in WordNet synsets source domain verbs are mixed with more abstract terms. For example, the metaphorical sense of *shape* in *shape opinion* is part of the synset (*de-*

termine, shape, mold, influence, regulate). This results in the baseline system tagging literal expressions as metaphorical, erroneously assuming that the verbs from the synset belong to the source domain.

The main source of confusion in the output of our clustering method was the conventionality of some metaphorical expressions, e.g. *hold views, adopt traditions, tackle a problem*. The system is capable of tracing metaphorical etymology of conventional phrases, but their senses are highly lexicalized. This lexicalization is reflected in the data and affects clustering in that conventional metaphors are sometimes clustered together with literally used terms, e.g. *tackle a problem* and *resolve a problem*, which may suggest that the latter are metaphorical. It should be noted, however, that such errors are rare.

Since there is no large metaphor-annotated corpus available, it was impossible for us to reliably evaluate the *recall* of the system. However, the system identified a total number of 4456 metaphorical expressions in the BNC starting with a seed set of only 62, which is a promising result.

5 Related Work

One of the first attempts to identify and interpret metaphorical expressions in text automatically is the approach of Fass (1991). Fass developed a system called met*, capable of discriminating between literalness, metonymy, metaphor and anomaly. It does this in three stages. First, literalness is distinguished from non-literalness using selectional preference violation as an indicator. In the case that non-literalness is detected, the respective phrase is tested for being a metonymic relation using hand-coded patterns (such as CONTAINER-for-CONTENT). If the system fails to recognize metonymy, it proceeds to search the knowledge base for a *relevant analogy* in order to discriminate metaphorical relations from anomalous ones. E.g., the sentence in (5) would be represented in this framework as (*car,drink,gasoline*), which does not satisfy the preference (*animal,drink,liquid*), as *car* is not a hyponym of *animal*. met* then searches its knowledge base for a triple containing a hypernym of both the actual argument and the desired argument and finds (*thing,use,energy_source*), which represents the metaphorical interpretation.

Birke and Sarkar (2006) present a sentence clustering approach for non-literal language recognition implemented in the TroFi system (Trope Finder). This idea originates from a similarity-based word sense disambiguation method developed by Karov and Edelman (1998). The method employs a set of seed sentences, where the senses are annotated, computes similarity between the sentence containing the word to be disambiguated and all of the seed sentences and selects the sense corresponding to the annotation in the most similar seed sentences. Birke and Sarkar (2006) adapt this algorithm to perform a two-way classification: literal vs. non-literal, and they do not clearly define the kinds of tropes they aim to discover. They attain a performance of 53.8% in terms of f-score.

The method of Gedigan et al. (2006) discriminates between literal and metaphorical use. They trained a maximum entropy classifier for this purpose. They obtained their data by extracting the lexical items whose frames are related to MOTION and CURE from FrameNet (Fillmore et al., 2003). Then they searched the PropBank Wall Street Journal corpus (Kingsbury and Palmer, 2002) for sentences containing such lexical items and annotated them with respect to metaphoricity. They used PropBank annotation (arguments and their semantic types) as features to train the classifier and report an accuracy of 95.12%. This result is, however, only a little higher than the performance of the naive baseline assigning majority class to all instances (92.90%). These numbers can be explained by the fact that 92.00% of the verbs of MOTION and CURE in the Wall Street Journal corpus are used metaphorically, thus making the dataset unbalanced with respect to the target categories and the task notably easier.

Both Birke and Sarkar (2006) and Gedigan et al. (2006) focus only on metaphors expressed by a verb. As opposed to that the approach of Krishnakumaran and Zhu (2007) deals with verbs, nouns and adjectives as parts of speech. They use hyponymy relation in WordNet and word bigram counts to predict metaphors at the sentence

level. Given an IS-A metaphor (e.g. *The world is a stage*[4]) they verify if the two nouns involved are in hyponymy relation in WordNet, and if this is not the case then this sentence is tagged as containing a metaphor. Along with this they consider expressions containing a verb or an adjective used metaphorically (e.g. *He planted good ideas in their minds* or *He has a fertile imagination*). Hereby they calculate bigram probabilities of verb-noun and adjective-noun pairs (including the hyponyms/hypernyms of the noun in question). If the combination is not observed in the data with sufficient frequency, the system tags the sentence containing it as metaphorical. This idea is a modification of the selectional preference view of Wilks. However, by using bigram counts over verb-noun pairs as opposed to verb-object relations extracted from parsed text Krishnakumaran and Zhu (2007) loose a great deal of information. The authors evaluated their system on a set of example sentences compiled from the Master Metaphor List (Lakoff et al., 1991), whereby highly conventionalized metaphors (they call them *dead metaphors*) are taken to be negative examples. Thus, they do not deal with literal examples as such: essentially, the distinction they are making is between the senses included in WordNet, even if they are conventional metaphors, and those not included in WordNet.

6 Conclusions and Future Directions

We presented a novel approach to metaphor identification in unrestricted text using unsupervised methods. Starting from a limited set of metaphorical seeds, the system is capable of capturing the regularities behind their production and annotating a much greater number and wider range of previously unseen metaphors in the BNC.

Our system is the first of its kind and it is capable of identifying metaphorical expressions with a high precision (0.79). By comparing its coverage to that of a WordNet baseline, we proved that our method goes far beyond synonymy and generalizes well over the source and target domains. Although at this stage we tested our system on verb-subject and verb-object metaphors only, we are convinced that the described identification techniques can be similarly applied to a wider range of syntactic constructions. Extending the system to deal with more parts of speech and types of phrases is part of our future work.

One possible limitation of our approach is that it is seed-dependent, which makes the recall of the system questionable. Thus, another important future research avenue is the creation of a more diverse seed set. We expect that a set of expressions representative of the whole variety of common metaphorical mappings, already described in linguistics literature, would enable the system to attain a very broad coverage of the corpus. Master Metaphor List (Lakoff et al., 1991) and other existing metaphor resources could be a sensible starting point on a route to such a dataset.

[4]William Shakespeare

Acknowledgments

We are very grateful to our anonymous reviewers for their useful feedback on this work and the volunteer annotators for their interest, time and help. This research is funded by generosity of Cambridge Overseas Trust (Katia Shutova), Dorothy Hodgkin Postgraduate Award (Lin Sun) and the Royal Society, UK (Anna Korhonen).

References

Andersen, O. E., J. Nioche, E. Briscoe, and J. Carroll. 2008. The BNC parsed with RASP4UIMA. In *Proceedings of LREC 2008*, Marrakech, Morocco.

Bergsma, S., D. Lin, and R. Goebel. 2008. Discriminative learning of selectional preference from unlabeled text. In *Proceedings of the EMNLP*.

Birke, J. and A. Sarkar. 2006. A clustering approach for the nearly unsupervised recognition of nonliteral language. In *In Proceedings of EACL-06*, pages 329–336.

Brew, C. and S. Schulte im Walde. 2002. Spectral clustering for German verbs. In *Proceedings of EMNLP*.

Briscoe, E., J. Carroll, and R. Watson. 2006. The second release of the rasp system. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 77–80.

Burnard, L. 2007. *Reference Guide for the British National Corpus (XML Edition)*.

Chen, J., D. Ji, C. Lim Tan, and Z. Niu. 2006. Unsupervised relation disambiguation using spectral clustering. In *Proceedings of COLING/ACL*.

Fass, D. 1991. met*: A method for discriminating metonymy and metaphor by computer. *Computational Linguistics*, 17(1):49–90.

Fellbaum, C., editor. 1998. *WordNet: An Electronic Lexical Database (ISBN: 0-262-06197-X)*. MIT Press, first edition.

Fillmore, C. J., C. R. Johnson, and M. R. L. Petruck. 2003. Background to FrameNet. *International Journal of Lexicography*, 16(3):235–250.

Gedigan, M., J. Bryant, S. Narayanan, and B. Ciric. 2006. Catching metaphors. In *In Proceedings of the 3rd Workshop on Scalable Natural Language Understanding*, pages 41–48, New York.

Joanis, E., S. Stevenson, and D. James. 2008. A general feature space for automatic verb classification. *Natural Language Engineering*, 14(3):337–367.

Karov, Y. and S. Edelman. 1998. Similarity-based word sense disambiguation. *Computational Linguistics*, 24(1):41–59.

Kingsbury, P. and M. Palmer. 2002. From TreeBank to PropBank. In *Proceedings of LREC-2002*, Gran Canaria, Canary Islands, Spain.

Kipper, K., A. Korhonen, N. Ryant, and M. Palmer. 2006. Extensive classifications of English verbs. In *Proceedings of the 12th EURALEX International Congress*.

Korhonen, A., Y. Krymolowski, and Z. Marx. 2003. Clustering polysemic subcategorization frame distributions semantically. In *Proceedings of ACL 2003*, Sapporo,Japan.

Korhonen, A., Y. Krymolowski, and T. Briscoe. 2006. A large subcategorization lexicon for natural language processing applications. In *Proceedings of LREC 2006*.

Krishnakumaran, S. and X. Zhu. 2007. Hunting elusive metaphors using lexical resources. In *Proceedings of the Workshop on Computational Approaches to Figurative Language*, pages 13–20, Rochester, NY.

Lakoff, G. and M. Johnson. 1980. *Metaphors We Live By*. University of Chicago Press, Chicago.

Lakoff, G., J. Espenson, and A. Schwartz. 1991. The master metaphor list. Technical report, University of California at Berkeley.

Levin, B. 1993. *English Verb Classes and Alternations*. University of Chicago Press, Chicago.

Lin, D. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th international conference on Computational linguistics*, pages 768–774.

Martin, J. H. 1988. Representing regularities in the metaphoric lexicon. In *Proceedings of the 12th conference on Computational linguistics*, pages 396–401.

Meila, M. and J. Shi. 2001. A random walks view of spectral segmentation. In *AISTATS*.

Meila, M. 2001. The multicut lemma. Technical report, University of Washington.

Pantel, P. and D. Lin. 2002. Discovering word senses from text. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 613–619. ACM.

Pereira, F., N. Tishby, and L. Lee. 1993. Distributional clustering of English words. In *Proceedings of ACL-93*, pages 183–190, Morristown, NJ, USA.

Pragglejaz Group. 2007. MIP: A method for identifying metaphorically used words in discourse. *Metaphor and Symbol*, 22:1–39.

Preiss, J., T. Briscoe, and A. Korhonen. 2007. A system for large-scale acquisition of verbal, nominal and adjectival subcategorization frames from corpora. In *Proceedings of ACL-2007*, volume 45, page 912.

Resnik, P. 1993. *Selection and Information: A Class-based Approach to Lexical Relationships*. Ph.D. thesis, Philadelphia, PA, USA.

Resnik, P. 1997. Selectional preference and sense disambiguation. In *ACL SIGLEX Workshop on Tagging Text with Lexical Semantics*, Washington, D.C.

Rooth, M., S. Riezler, D. Prescher, G. Carroll, and F. Beil. 1999. Inducing a semantically annotated lexicon via EM-based clustering. In *Proceedings of ACL 99*, pages 104–111.

Schulte im Walde, S. 2006. Experiments on the automatic induction of German semantic verb classes. *Computational Linguistics*, 32(2):159–194.

Shutova, E. and S. Teufel. 2010. Metaphor corpus annotated for source - target domain mappings. In *Proceedings of LREC 2010*, Malta.

Shutova, E. 2010. Automatic metaphor interpretation as a paraphrasing task. In *Proceedings of NAACL 2010*, Los Angeles, USA.

Siegel, S. and N. J. Castellan. 1988. *Nonparametric statistics for the behavioral sciences*. McGraw-Hill Book Company, New York, USA.

Explore the Structure of Social Tags by Subsumption Relations

Xiance Si, Zhiyuan Liu, Maosong Sun
Department of Computer Science and Technology
State Key Lab on Intelligent Technology and Systems
National Lab for Information Science and Technology
Tsinghua University
{sixiance,lzy.thu}@gmail.com, sms@tsinghua.edu.cn

Abstract

Thanks to its simplicity, social tagging system has accumulated huge amount of user contributed tags. However, user contributed tags lack explicit hierarchical structure, while many tag-based applications would benefit if such a structure presents. In this work, we explore the structure of tags with a directed and easy-to-evaluate relation, named as the subsumption relation. We propose three methods to discover the subsumption relation between tags. Specifically, the tagged document's content is used to find the relations, which leads to better result. Besides relation discovery, we also propose a greedy algorithm to eliminate the redundant relations by constructing a Layered Directed Acyclic Graph (Layered-DAG) of tags. We perform quantitative evaluations on two real world data sets. The results show that our methods outperform hierarchical clustering-based approach. Empirical study of the constructed Layered-DAG and error analysis are also provided.

1 Introduction

In this work, we aim at exploring the structure of social tags. Social tagging is widely used in Web-based services, in which a user could use any word to annotate an object. Thanks to its simplicity, services with social tagging features have attracted a lot of users and have accumulated huge amount of annotations. However, comparing to taxonomies, social tagging has an inherent shortcoming, that there is no explicit hierarchical relations between tags. Figure 1 (a) shows an example of the commonly used flat tag cloud, in which only the popularity of a tag is concerned. Kome et al. (2005) argued that implicit hierarchical relations exist in social tags. Previous literature shows that organizing tags in hierarchical structures will help tag-based Information Retrieval applications (Begelman et al., 2006; Brooks and Montanez, 2006).

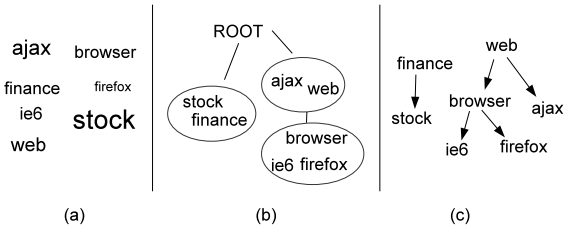


Figure 1: Examples of (a) flat tag cloud, (b) hierarchical clusters, and (c) subsumption relations.

Hierarchical clustering could reveal the similarity relations of tags. Figure 1 (b) shows an example of a typical hierarchical clustering of tags. While clusters can capture similarity between tags, problems still remain: First, clusters mix different relations, such as synonyms and hypernyms. Second, clusters also ignore the direction of relations, for example, the direction in `browser` $\rightarrow$ `firefox`. Third, it is hard to evaluate the correctness of clustering. Specifically, it is hard to tell if two tags are similar or not. In practice, directed and easy-to-evaluate relations between tags are preferred, such as Figure 1 (c).

In this work, we explore the structure of social tags by discovering a directed and easy-to-evaluate relation between tags, named *subsumption relation*. A tag t_a subsumes t_b, if and only if wherever t_b is used, we can also replace it

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1011–1019,
Beijing, August 2010

with t_a. Unlike *similar-to*, subsumption relation is asymmetric, and its correctness is easier to assess. Then, we propose three ways to discover the subsumption relations, through tag-tag, tag-word and tag-reason co-occurrences respectively. In the third way, A tag's *reason* is defined as the word in the content that explains the using of the tag. We employ the Tag Allocation Model (TAM) proposed by Si et al. (2010) to find the reason for each tag. Besides subsumption relation discovery, we also propose a greedy algorithm to remove the redundant relations. The removal is done by constructing a Layered Directed Acyclic Graph (Layered-DAG) of tags with the subsumption relations.

We carried out the experiments on two real world data sets. The results of quantitative evaluation showed that tag-reason based approach outperformed other two methods and a commonly used hierarchical clustering-based method. We also do empirical study on the output of Layered-DAG construction.

The contribution of this paper can be summarized as follows:

1. We explore the structure of social tags by a clearly defined subsumption relation. We propose methods to discover the subsumption relation automatically, leveraging both the co-occurred tags and the content of annotated document.
2. We propose an algorithm to eliminate the redundant relations by constructing a Layered-DAG of tags.
3. We perform both empirical and quantitative evaluation of proposed methods on two real world data sets.

The rest of the paper is organized as follows: Section 2 surveys the related work; Section 3 defines the subsumption relation we used, and proposes methods for relation discovery; Section 4 proposes a greedy algorithm for Layered-DAG construction; Section 5 explains the experimental settings and shows the evaluation results. Section 6 concludes the paper.

2 Related Work

To explore the hierarchical relations between tags, an intuitive way is to cluster the tags into hierarchical clusters. Wu et al. (2006b) used a factorized model, namely Latent Semantic Analysis, to group tags into non-hierarchical topics for better recommendation. Brooks et al. (2006) argued that performing Hierarchical Agglomerative Clustering (HAC) on tags can improve the collaborative tagging system. Later, HAC on tags was also used for improving personalized recommendation (Shepitsen et al., 2008). Heymann et al. (2006) clustered tags into a tree by a similarity-based greedy tree-growing method. They evaluated the obtained trees empirically, and reported that the method is simple yet powerful for organizing tags with hierarchies. Based on Heymann et al.'s work, Schwarzkopf et al. (2007) proposed an approach for modeling users with the hierarchy of tags. Begelman et al. (2006) used top-down hierarchical clustering, instead of bottom-up HAC, to organize tags, and argued that tag hierarchies improve user experiences in their system. Most of the hierarchical clustering algorithms rely on the symmetric similarity between tags, while the discovered relations are hard to evaluate quantitatively, since one cannot distinguish similar from not-similar with a clear boundary.

People have also worked on bridging social tagging systems and ontologies. An ontology defines relations between entities. Peter Mika (2005) proposed an extended scheme of social tagging that includes actors, concepts and objects, and used tag co-occurrences to construct an ontology from social tags. Wu et al. (2006a) used hierarchical clustering to build ontology from tags that also use similar-to relations. Later, ontology schemes that fits social tagging system were proposed, such as (Van Damme et al., 2007) and (Echarte et al., 2007), which mainly focused on the relation between tags, objects and users, rather than between tags themselves. Alexandre Passant (2007) mapped tags to domain ontologies manually to improve information retrieval in social media. To construct tag ontology automatically, Angeletou et al. (2007) used ontologies built by domain experts to find relations between tags, but observed a very low coverage. Specia et al. (2007) proposed an integrated framework for organizing tags by existing ontologies, but no experiment was performed. Kim et al. (2008) summarized the state-

of-the-art methods to model tags with semantic annotations.

Before social tagging was invented, Sanderson et al. (1999) proposed to use *subsumption* relation to organize words in text hierarchically. Schmitz et al. (2006) followed the idea to use subsumption relation for organizing Flickr [1] tag, where tag-tag co-occurrences are used for discover the relations. We follow the idea of subsumption relation in this paper, and explore alternative ways for relation discovery.

3 Subsumption Relations in Tags

In this section, we define the subsumption relation used in our study, and propose three methods to discover the subsumption relations.

3.1 Definitions

First, we introduce the symbols used through out the paper: A tag is denoted as $t \in T$, where T is the set of all tags. To distinguish from words, we use `fixed-width` to represent the example tags. An annotated document is denoted as $d \in D$, where D is the set of all documents. The words in d are denoted as a set $\{w_{d_i}\}$, where $i \in [1, |d|]$, and $|d|$ is the number of words in d.

Inspired by (Sanderson and Croft, 1999), we define the subsumption relation between t_a and t_b as follows: t_a **subsumes** t_b**, means that wherever the tag** t_b **is used,** t_a **can also be used without ambiguity**. The subsumption relation between t_a and t_b is denoted as $t_a \rightarrow_s t_b$.

Subsumption relation is directional, that is, $t_a \rightarrow_s t_b$ does not imply $t_b \rightarrow_s t_a$. For example, `literature` $\rightarrow_s$ `chineseliterature`, since for any document annotated with `chineseliterature`, we can also annotate it with `literature`. However, if we swapped the two tags, the statement would not hold.

Subsumption relation is more strict than similarity. For example, during the time of Haiti earthquake, the tag `earthquake` is close to `haiti` in similarity, but none of them implies the use of the other one: document annotated with `earthquake` may refer to the earthquake in China, while document annotated with `haiti` may mean the traveling experience in Haiti.

Note that the subsumption has transitivity property, that $t_a \rightarrow_s t_b$ and $t_b \rightarrow_s t_c$ means $t_a \rightarrow_s t_c$, which corresponds to our intuition. For instance, `naturaldisaster` $\rightarrow_s$ `earthquake` and `disaster` $\rightarrow_s$ `naturaldisaster` means `disaster` $\rightarrow_s$ `earthquake`.

3.2 Discover Subsumption Relation

We discover the subsumption relations by estimating the probability $p(t_a|t_b)$. The motivation is, if $t_a \rightarrow_s t_b$ and t_b is used, it would be more likely to see t_a. So, by sorting all (t_a, t_b) pairs by $p(t_a|t_b)$ in descending order, top-ranked pairs are more likely to have subsumption relations.

In this work, we present three methods to estimate the probability $p(t_a|t_b)$, using tag-tag, tag-word and tag-reason co-occurrences respectively. By using tag-word and tag-reason co-occurrences, we leverage the content of the annotated document for subsumption relation discovery.

3.2.1 Tag-Tag Co-occurrences Approach

The most intuitive way to estimate $p(t_a|t_b)$ is via tag-tag co-occurrences. Specifically, we use the following formula:

$$p(t_a|t_b) = \frac{N_d(t_a, t_b)}{N_d(t_b)}, \quad (1)$$

where $N_d(t_a, t_b)$ is the number of documents that are annotated by both t_a and t_b, and $N_d(t_b)$ is the number of documents annotated by t_b. We denote the tag-tag co-occurrences approach as TAG-TAG.

The use of TAG-TAG can be found in previous literature for organizing tags for photos(Schmitz, 2006). One of TAG-TAG's benefits is that it does not rely on the content of the annotated document, thus it can be applied to tags for non-text objects, such as images and music. However, when coming to text documents, this benefit is also a shortcoming, that TAG-TAG makes no use of the content when it is available.

Using TAG-TAG for subsumption relation discovery relies on an implication, that if a user has annotated d with t_b, he would also annotate all tags that subsumes t_b. The implication may not always hold in real world situations. For example,

[1] http://www.flickr.com. An image sharing site that allows users to annotate images with tags

a novel reader would use tags such as `scifi` and `mystery` to organize his collections, but he is not likely to annotate each of his collection as `novel` or `book`, since they are too obvious for him. We name the problem as the *omitted-tag problem*.

3.2.2 Tag-Word Co-occurrences Approach

When the content of the annotated document is available, using it for estimating $p(t_a|t_b)$ is a natural thought. The content is expected to be complete and information-rich whether or not the user has omitted any tags. We use the following formula to estimate $p(t_a|t_b)$ by tag-word co-occurrences:

$$\begin{aligned} p(t_a|t_b) &= \sum_{w \in W} p(t_a|w)p(w|t_b) \\ &= \sum_{w \in W} \frac{N_d(t_a, w)}{N_d(w)} \frac{N_d(t_b, w)}{N_d(t_b)}, \end{aligned} \quad (2)$$

where $N_d(t_a, w)$ is the number of documents that contains both tag t_a and word w, and $N_d(w)$ is the number of documents that contains the word w. We denote this approach as TAG-WORD.

Instead of computing tag-tag co-occurrences directly, TAG-WORD uses words in the document as a bridge to estimate $p(t_a|t_b)$. By introducing words, the estimation is less affected by the omitted-tag problem, Take the novel reader example again: Although he does not use the tag `novel` too often, the words in book descriptions would suggest the using of `novel`, according to all other documents annotated by `novel`.

While using the content may weaken the omitted-tag problem, it also brings the noise in text to the estimation. Not every word in the content is related to one of the tags. To the opposite, most words are functional words or that about other aspects of the document. $p(t_a|t_b)$ estimated by using all words may largely depends on these irrelevant words.

3.2.3 Tag-Reason Co-occurrences Approach

To focus on the words that are highly relevant to the interested tags, we propose the third method that uses tag-reason co-occurrences. The *reason* is defined as the word(s) that can explain the using of a tag in the document. For example, the tag `scifi` for a book could be explained by the words "robot", "Asimov" in the book description. If the reason of each tag could be identified, the noise in content-based $p(t_a|t_b)$ could be reduced.

Si et al. (2010) proposed a probabilistic model for content-based social tags, named Tag Allocation Model (TAM). TAM introduces a latent variable r for each tag in the data set, known as the reason variable. The value of r can be a word in the corresponding document, or a global noise variable μ. Allowing the reason of tags to be a global noise makes TAM deal with content-irrelevant tags and mistakenly annotated tags effectively. The likelihood that a document d is annotated by tag t is given as:

$$\begin{aligned} p(t|d) &= \sum_{w \in d} p(t|r = w)p(r = w|d)p(s = 0) \\ &+ p(t|\mu)p(r = \mu)p(s = 1), \end{aligned} \quad (3)$$

where r is the reason of the tag t, $r \in \{w_{di}|i \in [0, |d|]\} \cup \{\mu\}$, μ is the global noise variable. s is the source of reason t, $s = 0$ means the source is the content of the document, while $s = 1$ means the source is the global noise variable μ. TAM can be trained use Gibbs sampling method. For the details of TAM, please refer to (Si and Sun, 2010).

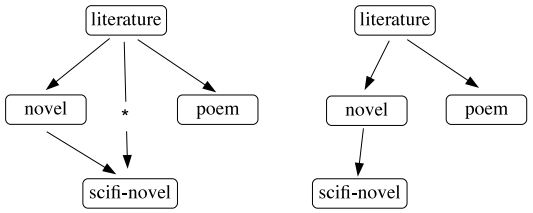
With a trained TAM, we can infer $p(t|r)$, the probability of seeing a tag t when using r as the reason, and $p(r|t)$, the probability of choosing r as the reason for tag t. With these probabilities, we can estimate $p(t_a|t_b)$ by

$$p(t_a|t_b) = \sum_{r \in W} p(t_a|r)p(r|t_b). \quad (4)$$

Note that we use only word reasons ($r \in W$), ignoring the noise reason μ completely. We denote this approach as TAG-REASON.

With the help of TAM, TAG-REASON covers the problems of the TAG-WORD method in two aspects: First, instead of using all words, TAG-REASON emphasizes on the really relevant words, which are the reasons identified by TAM. Second, by ignoring the noise variable μ, TAG-REASON is less affected by the content-irrelevant noise tags, such as `thingstodo` or `myown`.

After $p(t_a|t_b)$ is estimated for each $(t_a, t_b) \in T \times T$, we use the top-$n$ pairs with largest $p(t_a|t_b)$



(a) Directed Acyclic Graph (b) Layered Directed Acyclic Graph

Figure 2: DAG and Layered-DAG

as the final set of discovered subsumption relations.

4 Remove Redundancy with Layered-DAG Construction

The discovered subsumption relations connect all tags into a directed graph $G = \{V, E\}$, where V is the set of nodes, with each node is a tag; E is the set of edges, an edge e_{t_a,t_b} from t_a to t_b means $t_a \rightarrow_s t_b$. Furthermore, we define the weight of each edge w_e as the probability $p(t_a|t_b)$.

Recalling that subsumption relation has transitivity property, to avoid the cyclic references in G, we would like to turn G into a Directed Acyclic Graph (DAG). Further, DAG may also contains redundant information. Figure 2 (a) shows a part of a DAG. Note the edge marked as "*", which is perfectly correct, but does not provide extra information, since `literature` $\rightarrow_s$ `novel` and `novel` $\rightarrow_s$ `scifi-novel` have already implied that `literature` $\rightarrow_s$ `novel`. We would like to remove these redundant relations, turning a DAG into the form of Figure 2 (b).

We define Layered-DAG formally as follows: For a DAG G, when given any pair of nodes, if every path that can connect them has equal length, G is a Layered-DAG. Layered-DAG prohibits edges that link cross layers, such like edge "*" in Figure 2 (a). Constructing a Layered-DAG from the discovered relations can eliminate the redundant information.

Given a set of subsumption relations, multiple Layered-DAGs may be constructed. In particular, we want to find the Layered-DAG that maximizes the sum of all edges' weights. Weight maximization implies two concerns: First, when we need to remove a relation to resolve the conflicts or redundancy, the one with lower weight is prefered.

Layered-DAG Construction Algorithm

Input: A set of weighted relations, $R = \{t_a \rightarrow_s t_b | t_a \in T, t_b \in T\}$, $w_{t_a \rightarrow_s t_b} > 0$
Output: A Layered-DAG of tags $G^* = \{V^*, E^*\}$

1: $V^* = \{\}$
2: **while** $R \neq \varnothing$
3: **if** $V^* = \varnothing$
4: choose $t_a \rightarrow_s t_b \in R$ with highest weight.
5: $E^* \Leftarrow t_a \rightarrow_s t_b$
6: $V^* \Leftarrow t_a, V^* \leftarrow t_b$.
7: remove $t_a \rightarrow_s t_b$ from R.
8: **else**
9: $C \Leftarrow \{t_a \rightarrow_s t_b | t_a \rightarrow_s t_b \in R, \{t_a, t_b\} \cap V^* \neq \varnothing\}$
10: **for** $t_a \rightarrow_s t_b \in C$ in descending weight order
11: **if** adding $t_a \rightarrow_s t_b$ to G^* keeps G^* a Layered-DAG.
12: $E^* \Leftarrow t_a \rightarrow_s t_b$
13: $V^* \Leftarrow t_a, V^* \leftarrow t_b$.
14: **break**
15: **endif**
16: remove $t_a \rightarrow_s t_b$ from R.
17: **endfor**
18: **endif**
19: **endwhile**
20: **output** G^*

Figure 3: A greedy algorithm for constructing Layered-DAG of tags

Second, when more than one valid Layered-DAGs are available, we want to use the one that contains as many edges as possible.

Finding and proving an optimal algorithm for maximum Layered-DAG construction are beyond the scope of this paper. Here we present a greedy algorithm that works well in practice, as described in Figure 3.

The proposed algorithm starts with a minimal Layered-DAG G^* that contains only the highest weighted relation in R (Steps 1-8). Then, it moves an edge in G to G^* once a time, ensuring that adding the new edge still keeps G^* a valid Layered-DAG (Step 11), and the new edge has the highest weights among all valid candidates (Steps 9-10).

5 Experiments

In this section, we show the experimental results of proposed methods. Specifically, we focus on the following points:

- The quality of discovered subsumption relations by different methods.
- The characteristics of wrong subsumption relations discovered.
- The effect of Layered-DAG construction on the quality of relations.
- Empirical study of the resulted Layered-DAG.

Name	N	$\bar{N}_{tag}$	$\bar{N}_{content}$
BLOG	100,192	2.78	332.87
BOOK	110,371	8.51	204.76

Table 1: Statistics of the data sets. N is the number of documents. $\bar{N}_{tag}$ is the mean number of tags per document. $\bar{N}_{content}$ is the mean number of words per document.

5.1 Data Sets

We use two real world social tagging data sets. The first data set, named BLOG, is a collection of blog posts annotated by blog authors, which is crawled from the web. The second data set, named BOOK, is from a book collecting and sharing site[2], which contains description of Chinese books and user contributed tags. Table 1 lists the basic statistics of the data sets.

The two data sets have different characteristics. Documents in BLOG are longer, not well written, and the number of tags per document is small. To the opposite, documents in BOOK are shorter but well written, and there are more tags for each document.

5.2 Discovered Subsumption Relations

5.2.1 Experimental Settings

For BLOG, we use the tags that have been used more than 10 times; For BOOK, we use the tags that have been used more than 50 times. We perform 100 iterations of Gibbs sampling when training the TAM model, with first 50 iterations as the burn-in iterations. All the estimation methods require proper smoothing. Here we use additive smoothing for all methods, which adds a very small number (0.001 in our case) to all raw counts. Sophisticated smoothing method could be employed, but is out of the scope of this paper.

5.2.2 Evaluation

We use *precision* and *coverage* to evaluate the discovered relations at any given cut-off threshold n. First, we sort the discovered relations by their weights in descending order. Then, we take the top-n relations, discarding the others. For the remaining relations, precision is computed as N_c/n, N_c is the number of correct relations in the top-n list; coverage is computed as $N_t/|T|$, where N_t is the number of unique tags appeared in the top-n list, and $|T|$ is the total number of tags.

To get N_c, the number of correct relations, we need a standard judgement of the correctness of relations, which involves human labeling. To minimize the bias in human assessment, we use **pooling**, which is a widely accepted method in Information Retrieval research (Voorhees and Harman, 2005). Pooling works as follows: First, relations obtained by different methods are mixed together, creating a pool of relations. Second, the pool is shuffled, so that the labeler cannot identify the source of a single relation. Third, annotators are requested to label the relations in the pool as correct or incorrect, based on the definition of subsumption relation. After all relations in the pool are labeled, we use them as the standard judgement to evaluate each method's output.

Precision measures the proportion of correct relations, while coverage measures the proportion of tags that are connected by the relations. The cut-off threshold n affects both precision and coverage: the larger the n, the lower the precision, and the higher the coverage.

5.2.3 Baseline methods

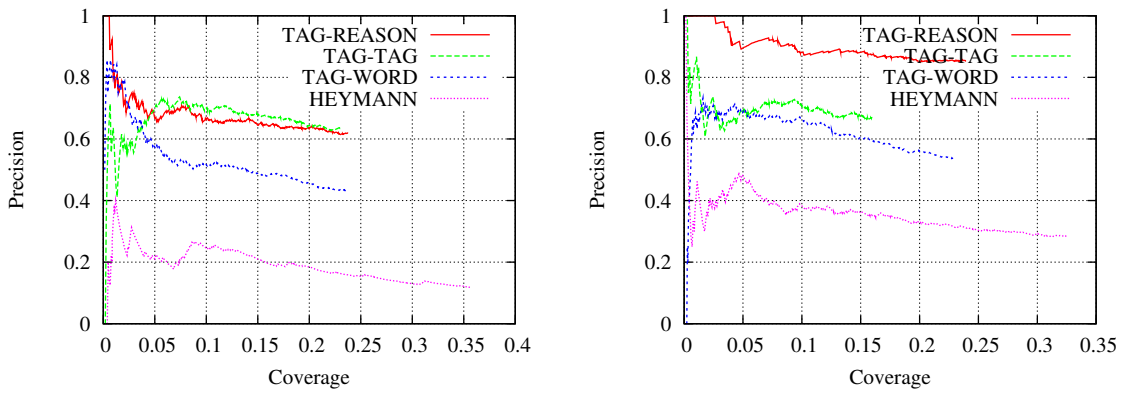
Besides TAG-TAG, TAG-WORD and TAG-REASON, we also include the method described in (Heymann and Garcia-Molina, 2006) as a baseline, denoted as HEYMANN. HEYMANN method was designed to find similar-to relation rather than subsumption relation. The similar-to relation is symmetric, while subsumption relation is more strict and asymmetric. In our experiments, we use the same evaluation process to evaluate TAG-TAG, TAG-WORD, TAG-REASON and HEYMANN, in which only subsumption relations will be marked as correct.

5.2.4 Results

For each method, we set the cut-off threshold n from 1 to 500, so as to plot the psrecision-coverage curves. The result is shown in Figure 4. The larger the area under the curve, the better the method's performance.

We have three observations from Figure 4. First, TAG-REASON has the best performance

[2]http://www.douban.com



(a) BLOG (b) BOOK

Figure 4: The precision and coverage of TAG-TAG, TAG-WORD, TAG-REASON and HEYMANN methods. The larger the area under the curve, the better the result. The cut-off threshold $n \in [1, 500]$.

BLOG			BOOK		
Insufficient	Reversed	Irrelevant	Insufficient	Reversed	Irrelevant
childedu $\rightarrow_s$ father	stock $\rightarrow_s$ security	travel$\rightarrow_s$building	textbook $\rightarrow_s$ exam	English $\rightarrow_s$ foreignlang	japan $\rightarrow_s$lightnovel
childedu $\rightarrow_s$ grandma	stock $\rightarrow_s$ financial	emotion $\rightarrow_s$time	history $\rightarrow_s$ military	biography$\rightarrow_s$people	building$\rightarrow_s$textbook
emotion$\rightarrow_s$warm	delicious$\rightarrow_s$taste	emotion$\rightarrow_s$original	piano$\rightarrow_s$scores	jpbuilding$\rightarrow_s$jpculture	sales$\rightarrow_s$O
childedu$\rightarrow_s$child	delicious$\rightarrow_s$food	culture$\rightarrow_s$spring	history$\rightarrow_s$culture	novel$\rightarrow_s$pureliterature	japan $\rightarrow_s$ shower
education$\rightarrow_s$child	earthquake$\rightarrow_s$disaster	poem$\rightarrow_s$night	novel$\rightarrow_s$love	ancientgreek$\rightarrow_s$greek	photo$\rightarrow_s$umbrella
Total 52%	Total 14%	Total 34%	Total 37%	Total 48%	Total 15%

Table 2: Examples of mistakes and the percentage of each mistake type.

on both data sets: On the BOOK data set, TAG-REASON outperforms others by a marked margin; On the BLOG data set, TAG-REASON has higher precision when coverage is smaller (which means within top-ranked relations), and has comparable precision to TAG-TAG when coverage increases. Second, similarity-based clustering method (namely HEYMANN) performed worse than others, suggesting it may not be adequate for discovering subsumption relation. Third, while also using content information, TAG-WORD performs poorer than both TAG-REASON and TAG-TAG, which suggests that noise in the content would prevent TAG-WORD from getting the correct estimation of $p(t_a|t_b)$.

To summarize, by leveraging *relevant* content, TAG-REASON could discover better subsumption relations than just using tag-tag co-occurrences and similarity-based hierarchical clustering.

5.2.5 Mistakes in Discovered Relations

We also studied the type of mistakes in subsumption relation discovery. To our observation, a mistakenly discovered relation $t_a \rightarrow_s t_b$ falls into one of the following categories:

1. **insufficient** t_a relates with t_b, but using t_b does not implies the using of t_a in all cases.
2. **reversed** $t_b \rightarrow_s t_a$ is correct, while $t_a \rightarrow_s t_b$ is not.
3. **irrelevant** There is no obvious connection between t_a and t_b.

We collected all incorrect relations discovered by the TAG-REASON method. Then, the type of mistake for each relation is labeled manually. The result is shown in Table 2, along with selected examples of each type.

Table 2 shows different error patterns for BLOG and BOOK. In BLOG, most of the mistakes are of the type *insufficient*. Taking "`education` $\rightarrow_s$ `child`" for example, annotating a document as `child` does not imply that it is about child education, it may about food or clothes for a child. In BOOK, most of the mistakes are *reversed* mistakes, which is a result of the omitted-tag problem discussed in Section 3.2.1.

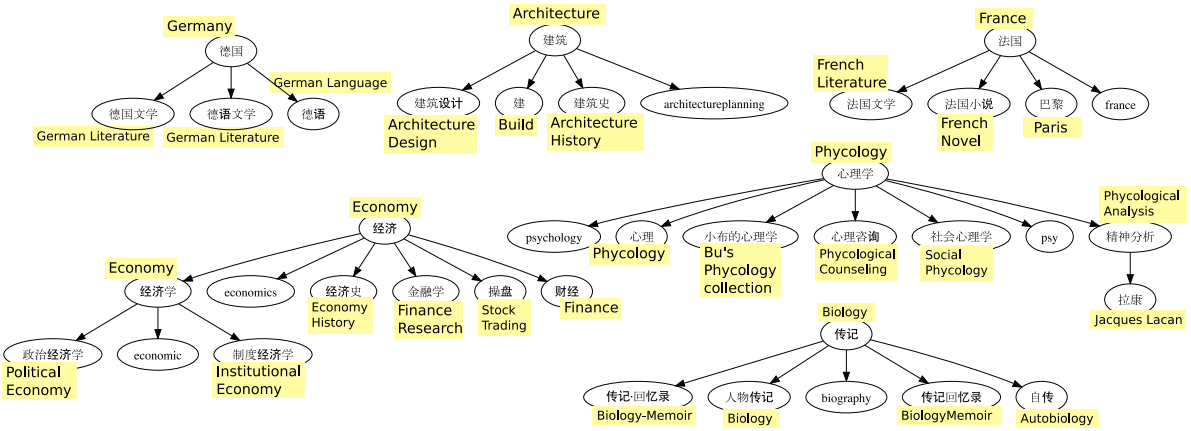


Figure 5: Part of the constructed Layered-DAG from the BOOK data set.

	BLOG		BOOK	
Method	Precision	Coverage	Precision	Coverage
TAG-TAG	-4.7%	$+7.9\%$	-7.4%	$+12.5\%$
TAG-WORD	0%	0%	-9.0%	$+2.2\%$
TAG-REASON	-3.6%	$+5.4\%$	-0.9%	$+5.4\%$

Table 3: The effects on precision and coverage by Layered-DAG construction

5.3 Layered-DAG Construction

Using the algorithm introduced in Section 4, we constructed Layered-DAGs from the discovered relations. Constructing Layered-DAG will remove certain relations, which will decrease the precision and increase the coverage. Table 3 shows the changes of precision and coverage brought by Layered-DAG construction. In most of the cases, the increasing of coverage is more than the decreasing of precision.

As a representative example, we show part of a constructed Layered-DAG from the BOOK data set in Figure 5, since the whole graph is too big to fit in the paper. All tags in Chinese are translated to English.

6 Conclusion and Future Work

In this paper, we explored the structure of social tags by discovering subsumption relations. First, we defined the subsumption relation $t_a \rightarrow_s t_b$ as t_a can be used to replace t_b without ambiguity. Then, we cast the subsumption relation identification problem to the estimation of $p(t_a|t_b)$. We proposed three methods, namely TAG-TAG, TAG-WORD and TAG-REASON, while the last two leverage the content of document to help estimation. We also proposed an greedy algorithm for constructing a Layered-DAG from the discovered relations, which helps minimizing redundancy.

We performed experiments on two real world data sets, and evaluated the discovered subsumption relations quantitatively by pooling. The results showed that the proposed methods outperform similarity-based hierarchical clusteing in finding subsumption relations. The TAG-REASON method, which uses only the relevant content to the tags, has the best performance. Empirical study showed that Layered-DAG construction works effectively as expected.

The results suggest two directions for future work: First, more ways for $p(t_a|t_b)$ estimation could be explored, for example, combining TAG-TAG and TAG-REASON; Second, external knowledge, such as the Wikipedia and the WordNet, could be exploited as background knowledge to improve the accuracy.

ACKNOWLEDGEMENTS

This work is supported by the National Science Foundation of China under Grant No. 60873174 and the National 863 High-Tech Program of China under Grant No. 2007AA01Z148. We also thank Douban Inc.(www.douban.com) for providing the DOUBAN data set, and Shoukun Wang, Guozhu Wen et al. of Douban Inc. for insightful discussion.

References

Angeletou, S., M. Sabou, L. Specia, and E. Motta. 2007. Bridging the gap between folksonomies and the semantic web: An experience report. In *Workshop: Bridging the Gap between Semantic Web and Web*, volume 2. Citeseer.

Begelman, Grigory, Keller, and F. Smadja. 2006. Automated tag clustering: Improving search and exploration in the tag space. In *Collaborative Web Tagging Workshop, 15 th International World Wide Web Conference*.

Brooks, Christopher H. and Nancy Montanez. 2006. Improved annotation of the blogosphere via autotagging and hierarchical clustering. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 625–632, New York, NY, USA. ACM.

Echarte, F., J. J. Astrain, A. Córdoba, and J. Villadangos. 2007. Ontology of folksonomy: A New modeling method. *Proceedings of Semantic Authoring, Annotation and Knowledge Markup (SAAKM)*.

Heymann, Paul and Hector Garcia-Molina. 2006. Collaborative creation of communal hierarchical taxonomies in social tagging systems. Technical Report 2006-10, Stanford University, April.

Kim, Hak L., Simon Scerri, John G. Breslin, Stefan Decker, and Hong G. Kim. 2008. The state of the art in tag ontologies: a semantic model for tagging and folksonomies. In *DCMI '08: Proceedings of the 2008 International Conference on Dublin Core and Metadata Applications*, pages 128–137. Dublin Core Metadata Initiative.

Kome, Sam H. 2005. Hierarchical subject relationships in folksonomies. Master's thesis, University of North Carolina at Chapel Hill, November.

Mika, P. 2005. Ontologies are us: A unified model of social networks and semantics. *The Semantic Web–ISWC 2005*, pages 522–536.

Passant, Alexandre. 2007. Using ontologies to strengthen folksonomies and enrich information retrieval in weblogs. In *Proceedings of International Conference on Weblogs and Social Media*.

Sanderson, M. and B. Croft. 1999. Deriving concept hierarchies from text. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 206–213. ACM.

Schmitz, P. 2006. Inducing ontology from flickr tags. In *Collaborative Web Tagging Workshop at WWW2006, Edinburgh, Scotland*, pages 210–214. Citeseer.

Schwarzkopf, E., D. Heckmann, and D. Dengler. 2007. In *Workshop on Data Mining for User Modeling, ICUM'07*, page 63. Citeseer.

Shepitsen, Andriy, Jonathan Gemmell, Bamshad Mobasher, and Robin Burke. 2008. Personalized recommendation in collaborative tagging systems using hierarchical clustering. In *Proceedings of ACM RecSys'08*.

Si, Xiance and Maosong Sun. 2010. Tag allocation model: Modeling noisy social annotations by reason finding. In *Proceedings of 2010 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*.

Specia, Lucia and Enrico Motta. 2007. Integrating folksonomies with the semantic web. pages 624–639.

Van Damme, C., M. Hepp, and K. Siorpaes. 2007. Folksontology: An integrated approach for turning folksonomies into ontologies. *Bridging the Gap between Semantic Web and Web*, 2:57–70.

Voorhees, E.M. and D.K. Harman. 2005. *TREC: Experiment and evaluation in information retrieval*. MIT Press.

Wu, Harris, Mohammad Zubair, and Kurt Maly. 2006a. Harvesting social knowledge from folksonomies. In *HYPERTEXT '06: Proceedings of the seventeenth conference on Hypertext and hypermedia*, pages 111–114, New York, NY, USA. ACM.

Wu, Xian, Lei Zhang, and Yong Yu. 2006b. Exploring social annotations for the semantic web. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 417–426, New York, NY, USA. ACM.

Ukwabelana - An open-source morphological Zulu corpus

Sebastian Spiegler
Intelligent Systems Group
University of Bristol
spiegler@cs.bris.ac.uk

Andrew van der Spuy
Linguistics Department
University of the Witwatersrand
andrew.vanderspuy@wits.ac.za

Peter A. Flach
Intelligent Systems Group
University of Bristol
peter.flach@bris.ac.uk

Abstract

Zulu is an indigenous language of South Africa, and one of the eleven official languages of that country. It is spoken by about 11 million speakers. Although it is similar in size to some Western languages, e.g. Swedish, it is considerably under-resourced. This paper presents a new open-source morphological corpus for Zulu named *Ukwabelana corpus*. We describe the agglutinating morphology of Zulu with its multiple prefixation and suffixation, and also introduce our labeling scheme. Further, the annotation process is described and all single resources are explained. These comprise a list of 10,000 labeled and 100,000 unlabeled word types, 3,000 part-of-speech (POS) tagged and 30,000 raw sentences as well as a morphological Zulu grammar, and a parsing algorithm which hypothesizes possible word roots and enumerates parses that conform to the Zulu grammar. We also provide a POS tagger which assigns the grammatical category to a morphologically analyzed word type. As it is hoped that the corpus and all resources will be of benefit to any person doing research on Zulu or on computer-aided analysis of languages, they will be made available in the public domain from http://www.cs.bris.ac.uk/Research/MachineLearning/Morphology/Resources/.

1 Introduction

Zulu (also known as isiZulu) is a Bantu language of South Africa, classified as S.30 in Guthrie's classification scheme (Guthrie, 1971). Since 1994, it has been recognized as one of the eleven official languages of South Africa. It has a written history of about 150 years: the first grammar was published by Grout (1859), and the first dictionary by Colenso (1905). There are about 11 million mother-tongue speakers, who constitute approximately 23% of South Africa's population, making Zulu the country's largest language.

Zulu is highly mutually intelligible with the Xhosa, Swati and Southern Ndebele languages, and with Ndebele of Zimbabwe (Lanham, 1960), to the extent that all of these can be considered dialects or varieties of a single language, Nguni. Despite its size, Zulu is considerably *under-resourced*, compared to Western languages with similar numbers of speakers, e.g. Swedish. There are only about four regular publications in Zulu, there are few published books, and the language is not used as a medium of instruction.

This of course is partly due to the short timespan of its written history, but the main reason, of course, is the apartheid history of South Africa: for most of the twentieth century resources were allocated to Afrikaans and English, the two former official languages, and relatively few resources to the indigenous Bantu languages. Since 1994, Zulu has had a much larger presence in the media, with several television programs being broadcast in Zulu every day. Yet much needs to be done in order to improve the resources available to Zulu speakers and students of Zulu.

The aim of the project reported in this paper was to establish a Zulu corpus, named the *Ukwabelana corpus*[1], consisting of morphologically labeled words (that is, word types) and part-of-speech (POS) tagged sentences. Along with the labeled corpus, unlabeled words and sentences, a morphological grammar, a semi-automatic mor-

[1] *Ukwabelana* means 'to share' in Zulu where the 'k' is pronounced voiced like a [g].

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1020–1028,
Beijing, August 2010

phological analyzer and a *POS tagger* for morphologically analyzed words will be provided.

The sources used for the corpus were limited to fictional works and the Zulu Bible. This means that there is not a wide variety of registers, and perhaps even of vocabulary items. This defect will have to be corrected in future work.

The *Ukwabelana corpus* can be used to develop and train automatic morphological analyzers, which in turn tag a large corpus of written Zulu, similar to the Brown corpus or the British National Corpus. Moreover, the list of POS tagged sentences is an essential step towards building an automatic syntactic tagger, which still does not exist for Zulu, and a tagged corpus of Zulu. Such a corpus would be beneficial to language researchers as it provides them with examples of actual usage, as opposed to elicited or invented examples, which may be artificial or unlikely to occur in real discourse. This would greatly improve the quality of Zulu dictionaries and grammars, most of which rely heavily on the work of Doke (1927) and Doke, Malcom and Sikakana (1958), with little in the way of innovation. Morphological tagging is also useful for practical computational applications like predictive text, spell-checking, grammar checking and machine translation; in the case of Zulu, where a large percentage of grammatical information is conveyed by prefixes and suffixes rather than by separate words, it is essential. For example, in English, the negative is expressed by means of a separate word *'not'*, but in Zulu the negative is constructed using a prefix-and-suffix combination on the verb, and this combination differs according to the mood of the verb (indicative, participial or subjunctive). The practical computational applications mentioned could have a very great impact on the use of Zulu as a written language, as spell-checking and grammar checking would benefit proofreaders, editors and writers. Machine translation could aid in increasing the number of texts available in Zulu, thus making it more of a literary language, and allowing it to become established as a language of education. The use of Zulu in public life could also increase. Currently, the tendency is to use English, as this is the language that reaches the widest audience. If high-quality automatic translation becomes available, this would no longer be necessary. As it is hoped that the *Ukwabelana corpus* will be of benefit to any person doing research on Zulu or on computer-aided analysis of languages, it will be made available as the first morphologically analysed corpus of Zulu in the public domain.

2 Related work

In this section, we will give an overview of linguistic research on Nguni languages, following the discussions in van der Spuy (2001), and thereafter a summary of computational approaches to the analysis of Zulu.

2.1 Linguistic research on Nguni languages

The five Nguni languages Zulu, Xhosa, South African Ndebele, Swati, and Zimbabwean Ndebele are highly mutually intelligible, and for this reason, works on any of the other Nguni languages are directly relevant to an analysis of Zulu.

There have been numerous studies of Nguni grammar, especially its morphology; in fact, the Nguni languages probably rival Swahili and Chewa for the title of most-studied Bantu language. The generative approach to morphological description (as developed by Aronoff (1976), Selkirk (1982), Lieber (1980), Lieber (1992)) has had very little influence on most of the work that has been done on Nguni morphology.

Usually, the descriptions have been atheoretical or structuralist. Doke's paradigmatic description of the morphology (Doke, 1927; Doke, 1935) has remained the basis for linguistic work in the Southern Bantu languages. Doke (1935) criticized previous writers on Bantu grammars for basing their classification, treatment and terminology on their own mother tongue or Latin. His intention was to create a grammatical structure for Bantu which did not conform to European or classical standards. Nevertheless, Doke himself could not shake off the European mindset: he treated the languages as if they had inflectional paradigms, with characteristics like subjunctive or indicative belonging to the whole word, rather than to identifiable affixes; in fact, he claimed (1950) that Bantu languages are "inflectional with [just] a tendency to agglutination", and assumed that the morphol-

ogy was linear not hierarchical. Most subsequent linguistic studies and reference grammars of the Southern Bantu languages have been directed at refining or redefining Doke's categories from a paradigmatic perspective.

Important Nguni examples are Van Eeden (1956), Van Wyk (1958), Beuchat (1966), Wilkes (1971), Nkabinde (1975), Cope (1984), Davey (1984), Louw (1984), Ziervogel et al. (1985), Gauton (1990), Gauton (1994), Khumalo (1992), Poulos and Msimang (1998), Posthumus (1987), Posthumus (1988), Posthumus (1988) and Posthumus (2000). Among the very few generative morphological descriptions of Nguni are Lanham (1971), Mbadi (1988) and Du Plessis (1993). Lanham (1971) gives a transformational analysis of Zulu adjectival and relative forms. This analysis can be viewed as diachronic rather than synchronic. Mbadi (1988) applies Lieber (1980) and Selkirk's percolation theory (Selkirk, 1982) to a few Xhosa morphological forms. Du Plessis (1993) gives a hierarchical description of the morphology of the verb, but he assumes that derivation is syntactical rather than lexical.

In short, there has been no thorough-going generative analysis of the morphology which has treated the Nguni languages as agglutinative rather than inflectional.

2.2 Computational approaches to analyzing Zulu

In the last decade, various computational approaches for Zulu have been reported. Based on the *Xerox finite-state toolbox* by Beesley and Karttunen (2003), Pretorius and Bosch (2003) developed a prototype of a computational morphological analyzer for Zulu. Using a semi-automated process, a morphological lexicon and a rule-base were built incrementally. Later work (Pretorius and Bosch, 2007) dealt with overgeneration of the Zulu finite-state tool concerning locative formation from nouns and verbal extensions to verb roots. Pretorius and Bosch (2009) also used cross-linguistic similarities and dissimilarities of Zulu to bootstrap a morphological analyser for Xhosa. Joubert et al. (2004) followed a bootstrapping approach to morphological analysis. A simple framework uses morpheme lists, morphophonological and morphosyntactic rules which are learnt by consulting an *oracle*, in their case a linguistic expert who corrects analyses. The framework then revises its grammar so that the updated morpheme lists and rules do not contradict previously found analyses. Botha and Barnard (2005) compared two approaches for gathering Zulu text corpora from the World Wide Web. They drew the conclusion that using commercial search engines for finding Zulu websites outperforms web-crawlers even with a carefully selected starting point. They saw the reason for that in the fact that most documents on the internet are in one of the world's dominant languages. Bosch and Eiselen (2005) presented a spell checker for Zulu based on morphological analysis and regular expressions. It was shown that after a certain threshold for the lexicon size performance could only be improved by incrementally extending morphological rules. Experiments were performed for basic and complex Zulu verbs and nouns, and large numbers of words still were not recognized. Spiegler et al. (2008) performed experiments where they tested four machine learning algorithms for morphological analysis with different degrees of supervision. An unsupervised algorithm analyzed a raw word list, two semi-supervised algorithms were provided with word stems and subsequently segmented prefix and suffix sequences, and the supervised algorithm used a language model of analysed words which was applied to new words. They experimentally showed that there is a certain trade-off between the usage of labeled data and performance. They also reckoned that computational analysis improves if words of different grammatical categories are analysed separately since there exist homographic morphemes across different word categories.

3 Zulu morphology

Zulu is an agglutinative language, with a complex morphology. It presents an especial problem for computational analysis, because words usually incorporate both prefixes and suffixes, and there can be several of each. This makes it hard to identify the root by mechanical means, as the root could be the first, second, third, or even a later morpheme in a word. The complexities involved are

exacerbated by the fact that a considerable number of affixes, especially prefixes, have allomorphic forms. This is largely brought about by the fact that Zulu has a prohibition against sequences of vowels, so that a prefix whose canonical form is *nga-* will have an allomorph *ng-* before roots that begin with vowels. Given a sequence *nga-*, then, it is possible that it constitutes an entire morpheme, or the beginning of a morpheme like the verb root *ngabaz-* 'to be uncertain', or a morpheme *ng-* followed by a vowel-commencing root like *and-* 'to increase'. Furthermore, many morphemes are homographs, so that the prefix *nga-* could represent either the potential mood morpheme or a form of the negative that occurs in subordinate clauses; and the sequence *ng-* could be the allomorph of either of these, or of a number of homographic morphemes *ngi-*, which represent the first person singular in various moods. Besides these phonologically conditioned allomorphs, there are also morphologically conditioned ones, for example the locative prefix *e-* has an allomorph *o-* that occurs in certain morphological circumstances. Certain morpheme sequences also exhibit syncretism, so that while most nouns take a sequence of prefixes known as the initial vowel and the noun prefix, as in *i-mi-zi* 'villages', nouns of certain classes, like class 5, syncretise these two prefixes, as in *i-gama* 'name', where the prefix *i-* represents both the initial vowel and the noun prefix.

Like all other Bantu languages, Zulu divides its nouns into a number of classes. The class is often identifiable from the noun prefix that is attached to the noun, and it governs the *agreement* of all words that modify the noun, as well as of predicates of which the noun is a subject. Object agreement may also be marked on the predicate. Two examples of this agreement are given below.

Example 1.

Leso si-tshudeni e-si-hle e-ngi-si-fundis-ile si-phas-e kahle.
that student who-AGR-good who-I-him-teach-PAST AGR-pass-PAST well.
'That good student whom I taught passed well.'

Example 2.

Lowo m-fundi o-mu-hle e-ngi-m-fundis-ile u-phas-e kahle.
that learner who-AGR-good who-I-him-teach-PAST AGR-pass-PAST well.
'That good learner whom I taught passed well.'

The differences in agreement morphology in the two sentences is brought about because the nouns *sitshudeni* and *mfundi* belong to different classes. Canonici (1996) argues that a noun should be assigned to a class by virtue of the agreement that it takes. In terms of this criterion, there are twelve noun classes in Zulu. These classes are numbered 1–7, 9, 10, 11, 14, 15. The numbering system was devised by Meinhof (1906), and reflects the historical affinities between Zulu and other Bantu languages: Zulu lacks classes 8, 12 and 13, which are found in other Bantu languages. In the labels used on the database, morphemes that command or show agreement have been labeled as *<xn>*, where *x* is a letter or sequence of letters, and *n* is a number: thus the morpheme *m-* in *mfundi* is labeled *<n1>*, as it marks the noun as belonging to noun class *1*. The morpheme *si-* in *engisifundisile* is marked *<o7>*, as it shows object agreement with a noun of class *7*.

Zulu *predicatives* may be either verbal or nonverbal – the latter are referred to in the literature as copulatives. Copulatives usually consist of a predicative prefix and a base, which may be a noun, an adjective, or a prepositional, locative or adverbial form. There may also be various tense, aspect and polarity markers. They translate the English verb 'be', plus its complement – Zulu has no direct equivalent of 'be'; the verb *-ba*, which has the closest meaning, is probably better translated as 'become'. Examples of copulative forms are *ubenguthisha* 'he was a teacher', *zimandla* 'they are strong', *basekhaya* 'they are at home'. Predicatives may occur in a variety of moods, tenses, aspects and polarities; these are usually distinguished by the affixes attached to the base form. Thus in *engasesendlini* '(s)he no longer being in the house', the initial prefix *e-* indicates third person singular, class 1, participial mood; the prefix *nga-* denotes negative; the first prefix *se-* denotes continuative aspect; the second prefix *se-* is the locative prefix; *n-* shows that the noun belongs to class 9; *dl-* is the noun root meaning 'house', an allomorph of the canonical form *-dlu*; and *-ini* is the locative suffix. Thus in typical agglutinative manner, each affix contributes a distinctive part of

the meaning of the word as a whole. This characteristic of the language was exploited in the labeling system used for the morphological corpus: labels were designed so as to indicate the grammatical function of the morpheme. A person searching for past tense negative verbs, for example, could simply search for the combination of *<past >*, *<neg>* and *<vr>*. A complete list of morphemes, allomorphs and their labels is provided along with the corpus and other resources.

According to the Dokean grammatical tradition (Doke, 1927), Zulu has a large number of parts of speech. This is because what would be separate words in other languages are often prefixes in Zulu, and also because various subtypes of determiner are given individual names. The parts of speech recognised in the corpus are: noun, verb, adjective, pronoun, adverb, conjunction, prepositional, possessive, locative, demonstrative, presentative, quantitative, copulative and relative.

Adjective includes the traditional Dokean adjective (a closed class of roots which take noun prefixes as their agreement prefixes) and the predicative form of the Dokean relative, which is seen as an open class of adjectives (cf. van der Spuy (2006)). *Pronouns* are the personal pronouns, which may also (sometimes in allomorphic form) be used as agreement morphemes in quantifiers. Adverbs may be forms derived from adjectives by prefixing *ka-* to the root, or morphologically unanalysable forms like *phansi* 'in front, forward'. Ideophones have been included as adverbs. *Prepositionals* are words that incorporate the Dokean "adverbials" *na-* 'with', *nga-* 'by means of', *njenga-* 'like', *kuna-* 'more than', etc., which are better analysed as prepositions. The presentative is Doke's "locative demonstrative copulative" - the briefer name was suggested by van der Spuy (2001). *Copulatives* are all Doke's copulatives, excluding the adjectives mentioned above. *Relatives* are all predicative forms incorporating a relative prefix.

4 The labeling scheme

The labeling scheme has been based on the idea that each morpheme in a word should be labeled, even when words belong to a very restricted class. For example, the demonstratives could have been labeled as composite forms, but instead it is assumed that demonstratives contain between one and three morphemes, e.g. *le<d>si<d7>ya<po3>* 'a demonstrative of the third position referring to class 7' - i.e.. 'that one yonder, class 7'. It should be possible from this detailed labeling to build up an amalgam of the morphological structure of the word. The labels have been chosen to be both as brief as possible and as transparent as possible, though transparency was often sacrificed for brevity. Thus indicative subject prefixes are labeled *<i1-15>*, relative prefixes are labeled *<r>*, and noun prefixes are labeled *<n1-15>*; but negative subject prefixes are labeled *<g1-15>* and possessive agreement prefixes are labeled *<z1-15>*. Sometimes a single label was used for several different forms, when these are orthographically distinct, so for example *<asp>* (aspect) is used as a label for the following, among others: the continuative prefix *sa-* and its allomorph *se-*, the exclusive prefix *se-*, and the potential prefix *nga-* and its allomorph *ng-*. A person searching for forms containing the potential aspect would have to search for *'nga<asp> + ng<asp>'*. However, there should be no ambiguity, as the orthographic form would eliminate this. The detailed description of the scheme is provided by Spiegler et al. (2010).

5 Annotation process

The goal of this project was to build a reasonably sized corpus of morphologically annotated words of high quality which could be later used for developing and training automatic morphological analyzers. For this reason, we had gathered a list of the commonest Zulu word types, defined a partial grammar and parsed Zulu words with a logic algorithm which proposes possible parses based on the partial grammar. Compared to a completely manual approach, this framework provided possible annotations to choose from or the option to type in an annotation if none of the suggestions was the correct one. This semi-automatic process speeded up the labeling by an estimated factor of 3-4, compared to a purely manual approach. In Figure 1 we illustrate the annotation process and in the following subsections each step is detailed.

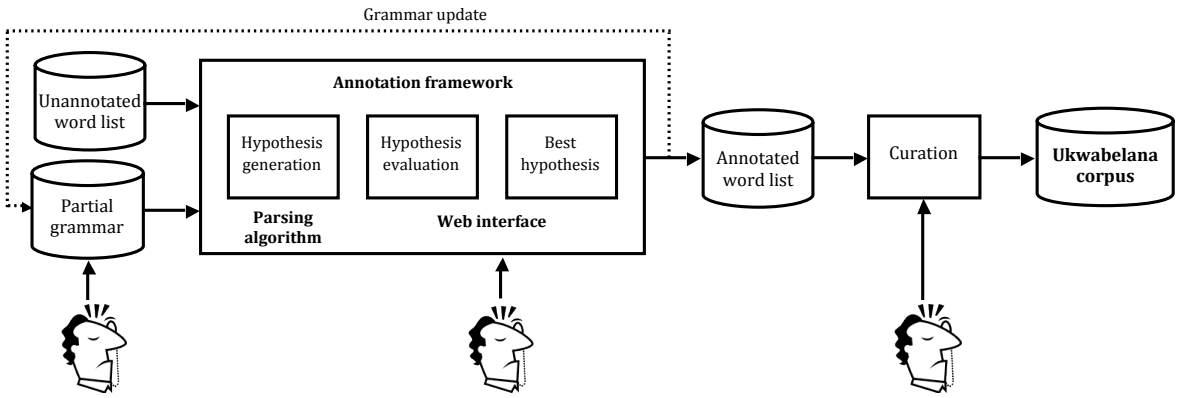


Figure 1: Process view of the annotation.

5.1 Unannotated word list

A list of unannotated Zulu words has been compiled from fictional works and the Zulu Bible. The original list comprises around 100,000 of the commonest Zulu word types. No information, morphological or syntactic, was given along with the words. We selected an initial subset of 10,000 words although our long-term goal is the complete analysis of the entire word list.

5.2 Partial grammar

Our choice for representing the morphological Zulu grammar was the formalism of *Definite Clause Grammars (DCGs)* used in the logic programming language *Prolog*. Although we defined our grammar as a simple context-free grammar, DCGs can also express context-sensitive grammars by associating variables as arguments to non-terminal symbols (Gazdar and Mellish, 1989). When defining our morphological grammar, we assumed that a linguistic expert could enumerate all or at least the most important morphological rules and morphemes of 'closed' morpheme categories, e.g. prefixes and suffixes of nouns and verbs. Morphemes of 'open' categories like noun and verb roots, however, would need to be hypothesized during the semi-automatic analysis and confirmed by the linguistic expert. Our final grammar comprised around 240 morphological rules and almost 300 entries in the morpheme dictionary. Since we did not only want to recognize admissible Zulu words but also obtain their morphological structure, we needed to extend our DCG by adding parse construction arguments as shown in the example below.

Example 3.

```
w((X)) --> n(X).
n((X,Y,Z)) --> iv(X),n2(Y),nr(Z).
iv(iv(a)) --> [a].
n2(n2(ba))--> [ba].
```

A possible parse for the word `abantu` 'people' could be `iv(a),n2(ba),*nr(ntu)` where '`*`' marks the hypothesized noun root.

With our partial grammar we could not directly use the inbuilt Prolog parser since we had to account for missing dictionary entries: Zulu verb and noun roots. We therefore implemented an algorithm which would generate hypotheses for possible parses according to our grammar. The algorithm will be described in the next subsection.

5.3 Hypothesis generation

For the hypothesis generation we reverted to logic programming and *abductive reasoning*. Abduction is a method of reasoning which is used with incomplete information. It generates possible hypotheses (parses) for an observation (word) and a given theory (grammar). Depending on the implementation, abduction finds the best hypothesis by evaluating all possible explanations. Our abductive algorithm is an extension of the meta-interpreter designed by Flach (1994) which only enumerates possible parses based on the grammar. A linguistic expert would then choose the best hypothesis. The algorithm invokes rules *top-down* starting with the most general until it reaches the last level of syntactic variables. These variables

are then matched against their dictionary entries from the left to the right of the word. A possible parse is found if either all syntactic variables can be matched to existing dictionary entries or if an unmatched variable is listed as *abducible*. Abducibles are predefined non-terminal symbols whose dictionary entry can be hypothesized. In our case, abducibles were noun and verb roots.

5.4 Evaluation and best hypothesis

Our annotation framework only enumerated allowable parses for a given word, therefore a linguistic expert needed to evaluate hypotheses. We provided a *web-interface* to the annotation framework, so that multiple users could participate in the annotation process. They would choose either a single or multiple correct parses. If none of the hypotheses were correct, the user would provide the correct analysis. Although our grammar was incomplete it still generated a substantial number of hypotheses per word. These were in no particular order and a result of the inherent ambiguity of Zulu morphology. We therefore experimented with different ways of improving the presentation of parses. The most promising approach was structural sorting. Parses were alphabetically re-ordered according to their morphemes and labels such that similar results were presented next to each other.

5.5 Grammar update

The grammar was defined in an iterative process and extended if the linguistic expert found morphemes of closed categories which had not been listed yet or certain patterns of incomplete or incorrect parses caused by either missing or inaccurate rules. The updated rules and dictionary were considered for newly parsed words.

5.6 Annotated word list and curation process

Although there had been great effort in improving the hypothesis generation of the parsing algorithm, a reasonable number of morphological analyses still had to be provided manually. During the curation process, we therefore had to deal with removing typos and standardizing morpheme labels provided by different experts. In order to guarantee a high quality of the morphological corpus, we also inspected single labels and analyses for their correctness. This was done by examining frequencies of labels and label combinations assuming that infrequent labels and combinations were likely to be incorrect and needed to be manually examined again. The finally curated corpus has an estimated error of 0.4 ± 0.5 incorrect single labels and 2.8 ± 2.1 incorrect complete analyses per 100 parses. Along with each word's analysis we wanted to provide part-of-speech (POS) tags. This was done by using a set of rules which determine the POS tag based on the morphological structure. We developed a prototype of a POS tagger which would assign the part-of-speech to a given morphological analysis based on a set of 34 rules. A summary of morphological analyses and words is given in Table 1. The rules are provided in Spiegler et al. (2010).

Category	**# Analyses**	**# Word types**
Verb	6965	4825
Noun	1437	1420
Relative	1042	988
Prepositional	969	951
Possessive	711	647
Copulative	558	545
Locative	380	379
Adverb	156	155
Modal	113	113
Demonstrative	63	61
Pronoun	38	31
Interjection	24	24
Presentative	15	15
Adjective	14	14
Conjunction	3	3
Total #	12488	10171

Table 1: Categories of labeled words.

5.7 POS tagging of sentences

In addition to the list of morphologically labeled words, we assigned parts-of-speech to a subset of 30,000 Zulu sentences. This task is straightforward if each word of a sentence only belongs to a single grammatical category. This was the case for 2595 sentences. For 431 sentences, however, we needed to disambiguate POS tags. We achieved this by analysing the left and right context of a word form and selecting the most probable part-of-speech from a given list of possible tags.

The overall error is estimated at 3.1 ± 0.3 incorrect POS tags per 100 words for the 3,000 sen-

Dataset	# Sentences	# Word tokens	#Word types	# Words per sentence	Word length
Raw	29,424	288,106	87,154	9.79±6.74	7.49±2.91
Tagged	3,026	21,416	7,858	7.08±3.75	6.81±2.68

Table 2: Statistics of raw and POS-tagged sentences.

tences we tagged. The summary statistics for raw and tagged sentences are shown in Table 2.

6 The Ukwabelana corpus - a resource description

The *Ukwabelana corpus* is three-fold:

1. It contains 10,000 morphologically labeled words and 3,000 POS-tagged sentences.
2. The corpus also comprises around 100,000 common Zulu word types and 30,000 Zulu sentences compiled from fictional works and the Zulu Bible, from which the labeled words and sentences have been sampled.
3. Furthermore, all software and additional data used during the annotation process is provided: the partial grammar in DCG format, the abductive algorithm for parsing with incomplete information and a prototype for a POS tagger which assigns word categories to morphologically analyzed words.

We are making these resources publicly available from `http://www.cs.bris.ac.uk/Research/MachineLearning/Morphology/Resources/` so that they will be of benefit to any person doing research on Zulu or on computer-aided analysis of languages.

7 Conclusions and future work

In this paper, we have given an overview of the morphology of the language Zulu, which is spoken by 23% and understood by more than half of the South African population. As an indigenous language with a written history of 150 years which was only recognised as an official languages in 1994, it is considerably under-resourced. We have spent considerable effort to compile the first open-source corpus of labeled and unlabeled words as well as POS-tagged and untagged sentences to promote research on this Bantu language. We have described the annotation process and the tools for compiling this corpus. We see this work as a first step in an ongoing effort to ultimately label the entire word and sentence corpus.

Our future work includes further automation of the annotation process by extending the described abductive algorithm with a more sophisticated hypothesis evaluation and by combining syntactical and morphological information during the decision process. Our research interest also lies in the field of automatic grammar induction which will help to refine our partial grammar. Another aspect is interactive labeling where a linguistic expert directs the search of an online parsing algorithm by providing additional information. Apart from the benefits to language researchers, we foresee an application of the corpus by machine learners which can develop and train their algorithms for morphological analysis.

Acknowledgements

We would like to thank Etienne Barnard and the Human Language Technologies Research Group from the Meraka Institute for their support during this project. Furthermore, we want to acknowledge Johannes Magwaza, Bruno Golénia, Ksenia Shalonova and Roger Tucker. The research work was sponsored by EPSRC grant EP/E010857/1 *Learning the morphology of complex synthetic languages* and a grant from the NRF (S. Africa).

References

Aronoff. 1976. *Word Formation in Generative Grammar*. The MIT Press.

Beesley and Karttunen. 2003. *Finite State Morphology*. University of Chicago Press.

Beuchat. 1966. The Verb in Zulu. *African Studies*, 22:137–169.

Bosch and Eiselen. 2005. The Effectiveness of Morphological Rules for an isiZulu Spelling Checker. *S. African Journal of African Lang.*, 25:25–36.

Botha and Barnard. 2005. Two Approaches to Gathering Text Corpora from the World Wide Web. *16th Ann. Symp. of the Pattern Recog. Ass. of S. Africa*.

Canonici. 1996. *Zulu Grammatical Structure.* Zulu Lang. and Literature, University of Natal, Durban.

Colenso. 1905. *Zulu-English Dictionary*. Natal, Vause, Slatter & Co.

Cope. 1984. An Outline of Zulu Grammars. *African Studies*, 43(2):83–102.

Davey. 1984. Adjectives and Relatives in Zulu. *S. African Journal of African Lang.*, 4:125–138.

Doke. 1927. *Text Book of Zulu Grammar*. Witwatersrand University Press.

Doke. 1935. *Bantu Linguistic Terminology*. Longman, Green and Co, London.

Doke. 1954. *Handbook of African Lang.*, chapter The S.ern Bantu Lang. Oxford University Press.

Doke, Malcom and Sikakana. 1958. *Zulu-English vocabulary*. Witwatersrand Uni. Press.

Du Plessis. 1993. *Linguistica: Festschrift EB van Wyk*, chapter Inflection in Syntax, pp. 61–66. Van Schaik, Pretoria.

Flach. 1994. *Simply Logical*. John Wiley.

Gauton. 1990. Adjektiewe en Relatiewe in Zulu. Master's thesis, University of Pretoria.

Gauton. 1994. Towards the Recognition of a Word Class 'adjective' for Zulu. *S. African Journal of African Lang.*, 14:62–71.

Gazdar and Mellish. 1989. *Natural Language Processing in Prolog*. Addison-Wesley.

Grout. 1859. *The Isizulu: A Grammar Of The Zulu Lang.* Kessinger Publishing.

Guthrie. 1971. *Comparative Bantu: An Introduction to the Comparative Linguistics and Prehistory of the Bantu Lang.* Farnborough, Gregg International Publishers.

Joubert, Zimu, Davel, and Barnard. 2004. A Framework for Bootstrapping Morphological Decomposition. Tech. report, CSIR/University of Pretoria, S. Africa.

Khumalo. 1992. *African Linguistic Contributions*, chapter The morphology of the direct relative in Zulu. Via Afrika.

Lanham. 1960. *The Comparative Phonology of Nguni*. Ph.D. thesis, Witwatersrand Uni., Jo'burg, S. Africa.

Lanham. 1971. The Noun as Deep-Structure Source for Nguni Adjectives and Relatives. *African Studies*, 30:294–311.

Lieber. 1980. *On the Organization of the Lexicon*. Ph.D. thesis, Massachusetts Institute of Technology.

Lieber. 1992. *Deconstructing Morphology*. The University of Chicago Press.

Louw. 1984. Word Categories in Southern Bantu. *African Studies*, 43(2):231–239.

Mbadi. 1988. *Anthology of Articles on African Linguistics and Literature*, chapter The Percolation Theory in Xhosa Morphology. Lexicon, Jo'burg.

Meinhof. 1906. *Grundzüge einer Vergleichenden Grammatik der Bantusprachen*. Reimer, Berlin.

Nkabinde. 1975. *A Revision of the Word Categories in Zulu*. Ph.D. thesis, University of S. Africa.

Posthumus. 1987. Relevancy and Applicability of Terminology Concerning the Essential Verb Categories in African Lang. *Logos*, 7:185–212.

Posthumus. 1988. Identifying Copulatives in Zulu and S.ern Sotho. *S. African Journal of African Lang.*, 8:61–64.

Posthumus. 2000. The So-Called Adjective in Zulu. *S. African Journal of African Lang.*, 20:148–158.

Poulos and Msimang. 1998. *A Linguistic Analysis of Zulu*. Via Afrika.

Pretorius and Bosch. 2003. Finite-State Computational Morphology: An Analyzer Prototype For Zulu. *Machine Translation*, 18:195–216.

Pretorius and Bosch. 2007. Containing Overgeneration in Zulu Computational Morphology. *Proceedings of 3rd Lang. and Technology Conference*, pp. 54 – 58, Poznan.

Pretorius and Bosch. 2009. Exploiting Cross-Linguistic Similarities in Zulu and Xhosa Computational Morphology. *Workshop on Lang. Technologies for African Lang. (AfLaT)*, pp. 96–103.

Selkirk. 1982. *The Syntax of Words*. MIT Press.

Spiegler, Golenia, Shalonova, Flach, and Tucker. 2008. Learning the Morphology of Zulu with Different Degrees of Supervision. *IEEE Workshop on Spoken Lang. Tech.*

Spiegler, van der Spuy, Flach. 2010. Additional material for the Ukwabelana Zulu corpus. Tech. report, University of Bristol, U.K.

van der Spuy. 2001. *Grammatical Structure and Zulu Morphology*. Ph.D. thesis, University of the Witwatersrand, Jo'burg, S. Africa.

van der Spuy. 2006. Wordhood in Zulu. *S.ern African Linguistics and Applied Lang. Studies*, 24(3):311–329.

Van Eeden. 1956. *Zoeloe-Grammatika*. Pro Ecclesia, Stellenbosch.

Van Wyk. 1958. *Woordverdeling in Noord-Sotho en Zulu: 'n bydrae tot die vraagstuk van word-identifikasie in die Bantoetale*. Ph.D. thesis, University of Pretoria.

Wilkes. 1971. *Agtervoegsels van die werkwoord in Zulu*. Ph.D. thesis, Rand Afrikaans University.

Ziervogel, Louw, and Taljaard. 1985. *A Handbook of the Zulu Lang.* Van Schaik, Pretoria.

EMMA: A Novel *E*valuation *M*etric for *M*orphological *A*nalysis

Sebastian Spiegler
Intelligent Systems Group
University of Bristol
spiegler@cs.bris.ac.uk

Christian Monson
Center for Spoken Language Understanding
Oregon Health & Science University
monsonc@csee.ogi.edu

Abstract

We present a novel *E*valuation *M*etric for *M*orphological *A*nalysis (*EMMA*) that is both linguistically appealing and empirically sound. *EMMA* uses a graph-based assignment algorithm, optimized via integer linear programming, to match morphemes of predicted word analyses to the analyses of a morphologically rich answer key. This is necessary especially for unsupervised morphology analysis systems which do not have access to linguistically motivated morpheme labels. Across 3 languages, *EMMA* scores of 14 systems have a substantially greater positive correlation with mean average precision in an information retrieval (IR) task than do scores from the metric currently used by the Morpho Challenge (MC) competition series. We compute *EMMA* and MC metric scores for 93 separate system-language pairs from the 2007, 2008, and 2009 MC competitions, demonstrating that *EMMA* is not susceptible to two types of gaming that have plagued recent MC competitions: Ambiguity Hijacking and Shared Morpheme Padding. The *EMMA* evaluation script is publicly available from http://www.cs.bris.ac.uk/Research/MachineLearning/Morphology/Resources/.

1 Introduction

Words in natural language are constructed from smaller building blocks called *morphemes*. For example, the word *wives* breaks down into an underlying stem, *wife*, together with a *plural* suffix. Analyzing the morphological structure of words is known to benefit a variety of downstream natural language (NL) tasks such as speech recognition (Creutz, 2006; Arısoy et al., 2009), machine translation (Oflazer et al., 2007), and information retrieval (McNamee et al., 2008).

A variety of automatic systems can morphologically analyze words that have been removed from their surrounding context. These systems range from hand-built finite state approaches (Beesley and Karttunen, 2003) to recently proposed algorithms which learn morphological structure in an unsupervised fashion (Kurimo et al., 2007). Since unsupervised systems do not have access to linguistically motivated morpheme labels, they typically produce morphological analyses that are closely related to the written form. Such a system might decompose *wives* as *wiv -es*. Meanwhile, a hand-built system might propose *wife_N +Plural*, or even parse *wives* as a hierarchical feature structure. As morphological analysis systems produce such varied outputs, comparing decompositions from disparate systems is a challenge.

This paper describes *EMMA*, an *E*valuation *M*etric for *M*orphological *A*nalysis that quantitatively measures the quality of a set of morphological analyses in a linguistically adequate, empirically useful, and novel fashion. *EMMA* evaluates analyses that can be represented as a flat set of symbolic features, including hierarchical representations, which can be projected down to a linearized form (Roark and Sproat, 2007).

An automatic metric that discriminates between proposed morphological analyses should

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1029–1037,
Beijing, August 2010

fulfill certain computational and linguistic criteria. Computationally, the metric should:

1. *Correlate* with the performance of real-world NL processing tasks which embed the morphological analyses.
2. Be *Readily Computable*: The metric will only be useful if it is less time consuming and easier to compute than the larger NL task.
3. Be *Robust*: The metric should be difficult to game and should accurately reflect the distribution of predicted and true morphemes.
4. Be *Readily Interpretable*: When possible, the final numeric score should directly identify the strengths and weaknesses of the underlying morphological analysis system.

While accounting for these computational requirements, a morphology metric should still reward accurate models of linguistic structure. In particular, the metric should account for:

1. *Morphophonology*: Applying a morphological rule may alter the surface form of stem or affix. In the word *wives*, /waivz/, a rule of morphophonology voices the stem-final /f/ of *wife*, /waif/, when the plural suffix is added. A metric should penalize for not placing *wives* and *wife* as forms of the same lexeme.
2. *Allomorphy*: A metric should capture the successful grouping of allomorphs. The German plural has several surface allomorphs including *-en* in *Zeiten* (*times*), *-e* in *Hunde* (*dogs*), and *-s* in *Autos* (*cars*). A metric should reward a morphological analysis system that analyzes the different surface forms of the German plural as underlyingly identical.
3. *Syncretism*: In mirror fashion, a metric should reward analyses that distinguish between surface-identical syncretic morphemes: although *derives* and *derivations* both contain an *-s* morpheme, one marks *3rd person singular* and the other *plural*.
4. *Ambiguity*: Finally, a metric should account for legitimate morphological ambiguity. In Hebrew, the written word *MHGR* has three viable morphological segmentations: *M- H- GR*, "*from the foreigner*", *M- HGR*, "*from Hagar*", and the unsegmented form *MHGR*, meaning "*immigrant*" (Lavie et al., 2004). Absent disambiguating context, a morphological system should be rewarded for calling out all three analyses for *MHGR*.

Morphophonology, allomorphy, syncretism, and ambiguity are all common phenomena in the world's languages. The first three have all received much discussion in theoretical linguistics (Spencer and Zwicky, 2001), while morphological ambiguity has significant practical implications in NL processing, e.g. in machine translation of morphologically complex languages (Lavie et al., 2004; Oflazer et al., 2007).

In Section 2 we propose the metric *EMMA*, which has been specifically designed to evaluate morphological analyses according to our computational and linguistic criteria. Section 3 then describes and qualitatively critiques several well-used alternative metrics. Section 4 empirically compares *EMMA* against the qualitatively-strong metric used in the Morpho Challenge competition series (Kurimo et al., 2009). And we conclude in Section 5.

2 *EMMA*: An *E*valuation *M*etric for *M*orphological *A*nalysis

EMMA, the metric we propose for the evaluation of morphological analyses, like all the metrics that we consider in this paper, compares proposed morphological analyses against an answer key of definitively-analyzed words from a vocabulary. Since a set of proposed analyses is likely to use a different labeling scheme than the answer key, especially true of the output from unsupervised systems, *EMMA* does not perform a direct comparison among proposed and answer analyses. Instead, *EMMA* seeks a one-to-one relabeling of the proposed morphemes that renders them as similar as possible to the answer key. *EMMA*, then, measures the degree to which proposed analyses approximate an isomorphism of the answer key analyses. For exposition, we initially assume that, for each word, a single proposed analysis is scored against a single unambiguous answer analysis. We relax this restriction in Section 2.3, where EMMA scores multiple proposed analyses

against a set of legitimately ambiguous morphological analyses.

To find the most appropriate one-to-one morpheme relabeling, *EMMA* turns to a standard algorithm from graph theory: optimal maximum matching in a bipartite graph. A *bipartite graph*, $G = \{X, Y; E\}$, consists of two disjoint sets of vertices, $X = \{x_1, x_2, \ldots, x_n\}$ and $Y = \{y_1, y_2, \ldots, y_m\}$, and a set of edges $e(x_i, y_j) \in E$ such that each edge has one end in X and the other end in Y. In *EMMA*, the set, A, of all unique morphemes in the answer key and the set, P, of all unique morphemes in the proposed analyses serve as the disjoint vertex sets of a bipartite graph.

A *matching* $M \subseteq E$ in a bipartite graph is defined as a set of edges $e(x_i, y_j)$ such that no x_i or y_j is repeated. A *maximum matching* is a matching where no M' with $|M'| > |M|$ exists. Furthermore, a weight $w(x_i, y_j) \in \Re$ may be assigned to each edge $e(x_i, y_j)$ of a bipartite graph. An *optimal assignment* is a maximum matching which also maximizes the sum of the weights of the edges of the matching

$$\sum_{e(x_i,y_j)\in M} w(x_i, y_j) \ .$$

EMMA weights the edge between a particular answer morpheme $a \in A$ and a proposed morpheme $p \in P$ as the number of words, w, in the vocabulary, V, where the answer analysis of w includes morpheme a while the proposed analysis includes p. *EMMA* constructs an optimal assignment maximum matching in this weighted bipartite morpheme graph. The edge weights ensure that the optimal matching will link the answer and proposed morphemes which globally occur in the analyses of the same words most often – restricting each answer morpheme to be represented by at most one proposed morpheme, and each proposed morpheme to represent at most one morpheme in the answer key. On the one hand, the restrictions thus imposed by bipartite matching penalize sets of proposed analyses that do not differentiate between surface-identical *syncretic morphemes*. On the other hand, the same one-to-one matching restrictions penalize proposed analyses that do not conflate *allomorphs* of the same underlying morpheme, whether those allomorphs are phonologically induced or not. Thus, *EMMA* meets our linguistic criteria from Section 1 of modeling syncretism, allomorphy, and morphophonology.

2.1 Maximum Matching by Integer Linear Programming

To construct the maximum matching optimal assignment of answer and proposed morphemes, *EMMA* uses standard integer linear programming techniques as implemented in *lpsolve* (Berkelaar et al., 2004). For the purpose of our integer program, we represent the weight of each potential edge of the optimal bipartite morpheme assignment in a count matrix $C = \{c_{ij}\}$ where c_{ij} is assigned the number of words $w \in V$ which share morpheme a_i in the answer key and p_j in the prediction. We then define a binary matrix $B = \{b_{ij}\}$ of the same dimensions as C. Each b_{ij} will be set to 1 if an edge exists from a_i to p_j in the optimal maximum matching, with $b_{ij} = 0$ otherwise. The integer linear program can then be defined as follows:

$$\underset{B}{\operatorname{argmax}} \sum_{i,j} (C \cdot B)_{ij} \tag{1}$$

$$s.t. \sum_i b_{ij} \leq 1 \ , \quad \sum_j b_{ij} \ \leq 1 \ , \quad b_{ij} \geq 0 \ ,$$

where $(C \cdot B)_{ij} = c_{ij} \cdot b_{ij}$ is the element-wise *Hadamard product*.

2.2 Performance Measures

Having settled on a maximum matching optimal assignment of proposed and answer morphemes, *EMMA* derives a final numeric score. Let w_k be the k^{th} word of V; and let A_k and P_k denote, respectively, the sets of morphemes in the answer key analysis of w_k and predicted analysis of w_k. Furthermore, let P_k^* denote the predicted morphemes for w_k where a morpheme p_j is replaced by a_i if $b_{ij} = 1$. Now that A_k and P_k^* contain morpheme labels that are directly comparable, we can define *precision* and *recall* scores for the proposed analysis of the word w_k. Precision is the fraction of correctly relabeled proposed morphemes from among all proposed morphemes of w_k; while *recall* is the number of correctly relabeled morphemes as a fraction of the answer key

analysis of w_k. Precision and recall of the full vocabulary are the average word-level precision and recall:

$$precision = \frac{1}{|V|} \sum_{k}^{|V|} \frac{|A_k \bigcap P_k^*|}{|P_k^*|} \quad , \qquad (2)$$

$$recall = \frac{1}{|V|} \sum_{k}^{|V|} \frac{|A_k \bigcap P_k^*|}{|A_k|} \quad . \qquad (3)$$

Finally, *f-measure* is the harmonic mean of precision and recall:

$$f\text{-}measure = \frac{2 \cdot precision \cdot recall}{precision + recall} \quad . \qquad (4)$$

2.3 Morphological Ambiguity in *EMMA*

Thus far we have presented *EMMA* for the scenario where each word has a single morphological analysis. But, as we saw in Section 1 with the Hebrew word *MHGR*, natural language permits surface forms to have multiple legitimate morphological analyses. When a word is truly ambiguous, *EMMA* expects an answer key to contain a set of analyses for that word. Similarly, we permit sets of proposed alternative analyses. To extend *EMMA* with the ability to evaluate alternative analyses we first generalize the optimal maximum matching of morphemes from Section 2.1. We then define a new integer linear program to match answer and proposed *alternative analyses*. Finally, we adjust the performance measures of Section 2.2 to account for alternatives.

2.3.1 Ambiguity and Morpheme Matching

Let $A_{k,r}$ denote the r^{th} alternative answer analysis of the k^{th} word with $1 \leq r \leq m_k$, and let $P_{k,s}$ denote the s^{th} alternative prediction with $1 \leq s \leq n_k$, where m_k is the number of alternative analyses in the answer key and n_k the number of alternative predictions for w_k. We redefine $A_k = \bigcup_r^{m_k} A_{k,r}$ and $P_k = \bigcup_s^{n_k} P_{k,s}$ as the set of all answer or, respectively, predicted morphemes of w_k across all analysis alternatives. Instead of incrementing each c_{ij} entry in the count matrix C by a full count, we now add $\frac{1}{m_k \cdot n_k}$ to c_{ij} for all pairs $(a_i, p_j) \in A_k \times P_k$. This corresponds to counting each combination of an answer key and predicted morpheme normalized by the number of possible pairings between proposed and answer analysis alternatives. When both the answer and proposed analyses consist of just a single alternative, c_{ij} remains unchanged. Generalized morpheme matching still employs the linear program defined in Equation 1.

2.3.2 Matching of Alternative Analyses

After performing a one-to-one morpheme relabelling that accounts for ambiguity, we need to extend *EMMA* with the ability to evaluate alternative analyses. We again turn to optimal maximum matching in a bipartite graph: Where earlier we matched proposed and answer morphemes, now we match full proposed and answer analysis alternatives, maximizing the total number of correctly predicted morphemes across all alternatives. Generalizing on the notation of the unambiguous case, let $P_{k,s}^*$ denote the s^{th} alternative predicted analysis of the k^{th} word where predicted morphemes have been replaced by their assigned answer key morphemes. We introduce a new count matrix $C' = \{c'_{r,s}\}$, where $c'_{r,s}$ is the count of common morphemes of the r^{th} answer key alternative and s^{th} predicted alternative. Based on Equation 1, we calculate the binary matrix $B' = \{b'_{r,s}\}$ which contains the optimal assignment of the alternative answer key and predicted analyses for w_k.

2.3.3 Ambiguity and Performance Scores

We now adjust *EMMA*'s numeric performance measures to account for sets of ambiguous analysis alternatives. *Precision* becomes

$$\frac{1}{|V|} \sum_{k}^{|V|} \frac{1}{n_k} \sum_{r}^{m_k} \sum_{s}^{n_k} b'_{r,s} \frac{|A_{k,r} \bigcap P_{k,s}^*|}{|P_{k,s}^*|} \quad , \qquad (5)$$

the ratio of correctly predicted morphemes across all predicted alternatives normalised by the number of predicted alternatives, n_k, and the vocabulary size, $|V|$. The factor $b'_{r,s}$ guarantees that scores are only averaged over pairs of proposed and answer analysis alternatives that have been assigned, that is, where $b'_{r,s} = 1$. *Recall* is measured similarly with

$$\frac{1}{|V|} \sum_{k}^{|V|} \frac{1}{m_k} \sum_{r}^{m_k} \sum_{s}^{n_k} b'_{r,s} \frac{|A_{k,r} \bigcap P_{k,s}^*|}{|A_{k,r}|} \quad . \qquad (6)$$

Here, we normalize by m_k, the number of alternative analyses for the k^{th} word that are listed in the answer key. The normalisation factors $\frac{1}{m_k}$ and $\frac{1}{n_k}$ ensure that predicting too few or many alternative analyses is penalised.

3 Other Morphology Metrics

Having presented the *EMMA* metric for evaluating the quality of a set of morphological analyses, we take a step back and examine other metrics that have been proposed. Morphology analysis metrics can be categorized as either: 1. Directly comparing proposed analyses against an answer key, or 2. Indirectly comparing proposed and answer analyses by measuring the strength of an isomorphic-like relationship between the proposed and answer morphemes. The proposed *EMMA* metric belongs to the second category of isomorphism-based metrics.

3.1 Metrics of Direct Inspection

By Segmentation Point. Perhaps the most readily accessible automatic evaluation metric is a direct comparison of the morpheme boundary positions in proposed and answer analyses. As early as 1974, Hafer and Weiss used the direct boundary metric. Although intuitively simple, the segmentation point method implicitly assumes that it is possible to arrive at a valid morphological analysis by merely dividing the characters of a word into letter sequences that can be reconcatenated to form the original word. But, by definition, concatenation cannot describe non-contanative processes like morphophonology and allomorphy. Nor does simple segmentation adequately differentiate between surface-identical syncretic morphemes. Despite these drawbacks, precision and recall of segmentation points is still used in current morphological analysis research (Poon et al. (2009), Snyder and Barzilay (2008), Kurimo et al. (2006)).

Against Full Analyses. To confront the reality of non-concatenative morphological processes, an answer key can hold full morphological analyses (as opposed to merely segmented surface forms). But while a hand-built (Beesley and Karttunen, 2003) or supervised (Wicentowski , 2002) morphology analysis system can directly model the annotation standards of a particular morphological answer key, the label given to specific morphemes is ultimately an arbitrary choice that an unsupervised morphology induction system has no way to discover.

By Hand. On the surface, scoring proposed analyses by hand appears to provide a way to evaluate the output of an unsupervised morphology analysis system. Hand evaluation, however, does not meet our criteria from Section 1 for a robust and readily computable metric. It is time consuming and, as Goldsmith (2001) explains, leaves difficult decisions of what constitutes a morpheme to on-the-fly subjective opinion.

3.2 Metrics of Isomorphic Analysis

Recognizing the drawbacks of direct evaluation, Schone and Jurafsky (2001), Snover et al. (2002), and Kurimo et al. (2007) propose related measures of morphological analysis quality that are based on the idea of an isomorphism. For reasons that will be clear momentarily, we refer to the Schone and Jurafsky, Snover et al., and Kurimo et al. metrics as *soft* isomorphic measures. As discussed in Section 2, metrics of isomorphism measure similarities between the distribution of proposed morphemes and the distribution of answer morphemes, where proposed and answer morphemes may be disjoint symbol sets.

Unlike the *EMMA* metric proposed in Section 2, the soft metrics of isomorphism do not seek to explicitly link proposed morphemes to answer morphemes. Instead, their metrics group sets or pairs of words which share, in *either* the proposed analyses or in the answer analyses, a stem (Schone and Jurafsky, 2001; Snover, 2002), a suffix (Snover et al., 2002), or any arbitrary morpheme (Kurimo et al., 2007). The soft metrics subsequently note whether these same sets or pairs of words share any morpheme in the answer key or, respectively, in the proposed analyses. By foregoing a hard morpheme assignment, the soft metrics do not adequately punish sets of proposed and answer morphemes which fail to model syncretism and/or allomorphy. For example, proposed analyses that annotate *3rd person singular* and *plural* with a single undifferentiated *+s* morpheme will receive recall credit for both nouns and

verbs.

3.3 The Morpho Challenge Metric

The Morpho Challenge (MC) competition series for unsupervised morphology analysis algorithms (Kurimo et al., 2009) has used a soft metric of isomorphism in its most recent three years of competition: 2007, 2008, and 2009. According to Kurimo et al. (2009) the Morpho Challenge (MC) measure samples *random word pairs* which share at least one common morpheme. Precision is calculated by generating random word pairs from the set of proposed analyses and then comparing the analyses of the word pairs in the answer key. The fraction of found and expected common morphemes is normalised by the number of words which are evaluated. *Recall* is defined in mirror fashion. The MC metric also normalizes precision and recall scores across sets of alternative analyses for each word in the proposal and answer key. To our knowledge the MC metric is the first isomorphism-based metric to attempt to account for *morphological ambiguity*. As we show in Section 4, however, MC's handling of ambiguity is easily gamed.

The MC metric does meet our criterion of being *readily computable* and, as we will show in the experimental section, the metric also *correlates* to a certain extent with performance on a higher-level natural language processing task. The downside of the MC metric, however, is *robustness*. In addition to MC's crude handling of ambiguity and its over-counting of allomorphs and syncretic morphemes, the random pair sampling method that MC uses is not independent of the set of analyses being evaluated. If two algorithms predict different morpheme distributions, the sampling method will find different numbers of word pairs. We substantiate our claim that the MC metric lacks robustness in Section 4 where we empirically compare it to the *EMMA* metric.

4 Experimental Evaluation

To experimentally evaluate our newly proposed *EMMA* metric, and to quantitatively compare the *EMMA* and MC metrics, we have evaluated results of 93 system-language pairs from Morpho Challenge 2007, 2008, and 2009.[1] The evaluation comprised three algorithms by Bernhard (2007) and Bernhard (2009), one algorithm by Can and Manandhar (2009), the MC baseline algorithm *Morfessor* by Creutz (2006), *UNGRADE* by Golenia et al. (2009), two algorithms by Lavallee and Langlais (2009), one algorithm by Lignos et al. (2009), five *ParaMor* versions by Monson et al. (2008) and Monson et al. (2009), three *Promodes* versions by Spiegler et al. (2009) and one algorithm by Tchoukalov et al. (2009). We ran these algorithms over six data sets available from the MC competition: Arabic (vowelized and non-vowelized), English, Finnish, German, and Turkish. We then scored the system outputs using both *EMMA* and the MC metric against an answer key provided by MC. In Sections 2 and 3.3 we have already commented on the *linguistic characteristics* of both metrics. In this section, we concentrate on their *computational performance*.

Both the *EMMA* and MC metrics are *readily computable*: Both are freely available[2] and they each take less than two minutes to run on the average desktop machines we have used. In terms of *interpretability*, *EMMA* not only returns the performance as precision, recall and f-measure as MC does, but also provides predicted analyses where mapped morphemes are replaced by answer key morphemes. This information is helpful when judging results qualitatively since it exposes tangible algorithmic characteristics. In Table 1 we present the algorithms with the highest MC and *EMMA* scores for each language. For all languages, the *EMMA* and MC metrics place different algorithms highest. One reason for the significantly different rankings that the two metrics provide may be the *sampling of random pairs* that MC uses. Depending on the distribution of predicted morphemes across words, the number of random pairs, which is used for calculating the precision, may vary. For instance, on vowelized Arabic, *Promodes 1* is evaluated over a sample of 100 pairs where MC selected just 47 pairs for *ParaMor Mimic*.

[1] Detailed results can be found in Spiegler (2010).

[2] *EMMA* may be downloaded from `http://www.cs.bris.ac.uk/Research/MachineLearning/Morphology/Resources/`

Language	Algorithm and year of participation in MC		MC evaluation metric			*EMMA* evaluation metric		
			Pr.	Re.	F1	Pr.	Re.	F1
Arabic (nv)	Promodes 2	2009	0.7789	0.3980	**0.5268**	0.5356	0.2444	0.3356
	Ungrade	2009	0.7971	0.1603	0.2670	0.7017	0.2490	**0.3675**
Arabic (vw)	Promodes 2	2009	0.5946	0.6017	**0.5982**	0.4051	0.3199	0.3575
	Promodes 1	2009	0.7381	0.3477	0.4727	0.5588	0.3281	**0.4135**
English	Bernhard 1	2007	0.7850	0.5763	**0.6647**	0.8029	0.7460	0.7734
	Lignos	2009	0.7446	0.4716	0.5775	0.9146	0.6747	**0.7766**
Finnish	ParaMorPlusMorfessor	2008	0.5928	0.5675	**0.5798**	0.2271	0.3428	0.2732
	Lavallee rali-cof	2009	0.6731	0.3563	0.4659	0.5061	0.4065	**0.4509**
German	ParaMorPlusMorfessor	2008	0.5562	0.6077	**0.5808**	0.3633	0.4948	0.4190
	Morfessor	2009	0.6528	0.3818	0.4818	0.7311	0.5556	**0.6314**
Turkish	ParaMorPlusMorfessor	2008	0.6779	0.5732	**0.6212**	0.3476	0.4315	0.3851
	Morfessor	2009	0.7894	0.3330	0.4684	0.5901	0.3703	**0.4550**

Table 1: Best performing algorithms with MC and *EMMA* evaluation metric.

Algorithm and year of participation in MC		MC evaluation metric			*EMMA* evaluation metric		
		Pr.	Re.	F1	Pr.	Re.	F1
Morfessor	2009	0.8143	0.2788	0.4154	0.4751	0.3472	0.4012
ParaMor	2008	0.4111	0.4337	0.4221	0.4322	0.3770	0.4027
ParaMorPlusMorfessor	2008	0.5928	0.5675	**0.5798**	0.2271	0.3428	**0.2732**
Paramor Morfessor Union	2009	0.4374	0.5676	0.4941	0.3878	0.4530	0.4178

Table 3: Gaming MC with ambiguity hijacking on Finnish.

Looking at any particular algorithm-language pair, the *EMMA* and MC scores differ considerably and respective raw scores are not directly comparable. More interesting is the extent to which both metrics *correlate* with real NL tasks. Table 2 lists the Spearman rank correlation coefficient for algorithms from MC 2009 on English, Finnish and German comparing rankings of f-measure results returned by either MC or *EMMA* against rankings using the mean average precision (MAP) of an information retrieval (IR) task.[3] All MAP scores are taken from Kurimo et al. (2009). Although both metrics positively correlate with the IR results; *EMMA*'s correlation is clearly stronger across all three languages.

To test the *robustness* of the *EMMA* and MC metrics, we performed two experiments where we intentionally attempt to game the metrics – *ambiguity hijacking* and *shared morpheme padding*. In both experiments, the MC metric showed vulnerability. Ambiguity hijacking results for Finnish appear in Table 3, other languages perform similarly. Using both metrics, we scored the Finnish analyses that were proposed by a) the *Morfessor* algorithm alone, b) *ParaMor* alone, and c) two ways of combining ParaMor and Morfessor: *ParaMorPlusMorfessor* simply lists the ParaMor and Morfessor analyses as alternatives – as if each word were ambiguous between a ParaMor and a Morfessor analysis; *ParaMorMorfessorUnion*, on the other hand, combines the morpheme boundary predictions of *ParaMor* and *Morfessor* into a single analysis. The *ParaMorPlusMorfessor* system games the ambiguity mechanism of the MC metric, achieving an f-measure higher than that of any of the three other algorithms. *EMMA*, however, correctly discovers that the analyses proposed by *ParaMorPlusMorfessor* lie farther from an isomorphism to the the answer key than do the unified analyses of *ParaMorMorfessorUnion*.

In Table 4 we show a second way of gaming the MC metric – *shared morpheme padding*. We add the same unique bogus morpheme to each proposed analysis of every word for all systems.

[3] Detailed results can be found in Spiegler (2010).

Language	MC evaluation			EMMA evaluation		
	Precision	Recall	F-measure	Precision	Recall	F-measure
Arabic (nv)	0.91±0.02	**10.83 ± 8.33**	**7.20±5.10**	0.91±0.05	1.30±0.07	1.20±0.05
Arabic (vw)	0.85±0.04	**11.17±8.81**	**7.13±5.23**	0.89±0.07	1.21±0.06	1.12±0.05
English	**0.36±0.08**	**2.02±0.66**	0.63±0.10	0.73±0.15	1.05±0.08	0.86±0.12
Finnish	0.57±0.08	**3.07±2.47**	1.19±0.68	0.87±0.19	1.12±0.10	0.99±0.14
German	**0.43±0.08**	**2.90±1.45**	0.84±0.16	0.80±0.17	1.09±0.08	0.94±0.11
Turkish	0.58±0.09	**2.95±1.65**	1.19±0.37	0.85±0.08	1.07±0.04	0.97±0.05

Table 4: Gaming MC with shared morpheme padding: Average and standard deviations of the ratio of padded to original scores.

Padding analyses with a shared morpheme significantly increases the recall scores of the MC metric. We summarize our experimental results by calculating, for each language-algorithm pair, the ratio of the score for the padded analyses as compared to that of the original, unpadded analyses. Table 4 reports average and standard deviation of the ratios across all systems for each language. In Arabic (nv. and vw.), the recall increases by 10.83 and 11.17 times, which leads to an inflation of f-measure by 7.20 and 7.13 times – this is a direct result of the soft nature of the MC isomorphism. In contrast, *EMMA*'s recall scores increase much less than MC's do, and *EMMA*'s precision scores decrease proportionately. A small change to the set of proposed analyses does not lead to a huge difference in f-measure – characteristic of a more *robust* metric.

5 Conclusion

This paper has proposed, *EMMA*, a novel evaluation metric for the assessment of the quality of a set of morphological analyses. *EMMA*'s:

1. Coverage of the major morphological phenomena,
2. Correlation with performance on natural language processing tasks, and
3. Computational robustness

all recommend the the metric as a strong and useful measure – particularly when evaluating unsupervised morphology analysis systems which, lacking access to labeled training data, are uninformed of the labeling standard used in the answer key.

	Correlation with IR	
	IR vs. MC	IR vs. *EMMA*
English	0.466	**0.608**
Finnish	0.681	**0.759**
German	0.379	**0.637**

Table 2: Spearman rank correlation coefficient of metrics vs. Information Retrieval (IR).

Acknowledgements

We would like to acknowledge various fruitful discussions with Aram Harrow, Alex Popa, Tilo Burghardt and Peter Flach. The work was partially sponsored by EPSRC grant EP/E010857/1 *Learning the morphology of complex synthetic languages*, as well as by NSF Grant #IIS-0811745 and DOD/NGIA grant #HM1582-08-1-0038.

References

Arısoy, Ebru, Doğan Can, Sıddıka Parlak, Haşim Sak, and Murat Saraçlar. 2009. Turkish Broadcast News Transcription and Retrieval. *IEEE Trans. on Audio, Speech and Lang. Proc.*

Beesley, Kenneth R. and Lauri Karttunen. 2003. *Finite State Morphology*. University of Chicago Press.

Berkelaar, Michel, Kjell Eikland, and Peter Notebaert. 2004. Open source (mixed-integer) linear programming system, version 5.1.0.0. `http://lpsolve.sourceforge.net/`.

Bernhard, Delphine. 2007. Simple morpheme labelling in unsupervised morpheme analysis. *Working Notes, CLEF 2007 Workshop*.

Bernhard, Delphine. 2009. Morphonet: Exploring the use of community structure for unsupervised morpheme analysis. *Working Notes, CLEF 2009 Workshop*.

Can, Burcu and Suresh Manandhar. 2009. Unsupervised learning of morphology by using syntactic categories. *Working Notes, CLEF 2009 Workshop*.

Creutz, Mathias. 2006. *Induction of the Morphology of Natural Language: Unsupervised Morpheme Segmentation with Application to Automatic Speech Recognition*. Ph.D. thesis, Helsinki University of Technology, Espoo, Finland.

Goldsmith, John. 2001. Unsupervised learning of the morphology of a natural language. *Comp. Ling.*, 27.

Golénia, Bruno, Sebastian Spiegler, and Peter Flach. 2009. Ungrade: unsupervised graph decomposition. *Working Notes, CLEF 2009 Workshop*.

Hafer, M. A. and S. F. Weiss. 1974. Word segmentation by letter successor varieties. *Inf. Storage and Retrieval*, 10.

Kurimo, Mikko, Mathias Creutz, Matti Varjokallio, Ebru Arisoy, Murat Saraclar. 2006. Unsupervised segmentation of words into morphemes - Morpho Challenge 2005. *Interspeech*.

Kurimo, Mikko, Mathias Creutz, and Ville Turunen. 2007. Overview of morpho challenge in CLEF 2007. *Working Notes, CLEF 2007 Workshop*.

Kurimo, Mikko and Ville Turunen. 2008. Unsupervised Morpheme Analysis Evaluation by IR experiments – Morpho Challenge 2008. *Working Notes, CLEF 2008 Workshop*.

Kurimo, Mikko, Sami Virpioja, and Ville T. Turunen. 2009. Overview and results of morpho challenge 2009. *Working Notes, CLEF 2009 Workshop*.

Lavallee, Jean-Francois and Philippe Langlais. 2009. Morphological Acquisition by Formal Analogy. *Working Notes, CLEF 2009 Workshop*.

Lavie, Alon, Erik Peterson, Katharina Probst, Shuly Wintner, Yaniv Eytani. 2004. Rapid Prototyping of a Transfer-based Hebrew-to-English Machine Translation System. *Proc. of TMI-2004*.

Lignos, Constantine, Erwin Chan, Mitchell P. Marcus, and Charles Yang. 2009. A rule-based unsupervised morphology learning framework. *Working Notes, CLEF 2009 Workshop*.

McNamee, Paul, Charles Nicholas, and James Mayfield. 2008. Don't Have a Stemmer? Be Un+concern+ed *Proc. of the 31st Anual International ACM SIGIR Conference 20-24 July 2008*.

Monson, Christian, Jaime Carbonell, Alon Lavie, and Lori Levin. 2008. Paramor and morpho challenge 2008. *Working Notes, CLEF 2008 Workshop*.

Monson, Christian, Kristy Hollingshead, and Brian Roark. 2009. Probabilistic paramor. *Working Notes, CLEF 2009 Workshop*.

Oflazer, Kemal, and İlknur Durgar El-Kahlout. 2007. Different Representational Units in English-to-Turkish Statistical Machine Translation. *Proc. of Statistical Machine Translation Workshop at ACL 2007*.

Poon, Hoifung, Colin Cherry and Kristina Toutanova 2009. Unsupervised Morphological Segmentation with Log-Linear Models. *Proc. of ACL*.

Roark, Brian and Richard Sproat. 2007. *Computational Approaches to Morphology and Syntax*. Oxford Univ. Press.

Schone, Patrick and Daniel Jurafsky. 2001. Knowlege-free induction of inflectional morphologies. *Proc. of NAACL-2001*.

Snover, Matthew G., Gaja E. Jarosz and Michael R. Brent. 2002. Unsupervised Learning of Morphology Using a Novel Directed Search Algorithm: Taking the First Step. *Proc. of the ACL-02 SIGPHON Workshop*.

Snyder, Benjamin and Regina Barzilay. 2008. Unsupervised Multilingual Learning for Morphological Segmentation. *Proc. of ACL-08: HLT*.

Spencer, Andrew and Arnold M. Zwicky, editors. 2001. *The Handbook of Morphology*. Wiley-Blackwell.

Spiegler, Sebastian, Bruno Golénia, and Peter A. Flach. 2009. Promodes: A probabilistic generative model for word decomposition. *Working Notes, CLEF 2009 Workshop*.

Spiegler, Sebastian. 2010. *EMMA: A Novel Metric for Morphological Analysis - Experimental Results in Detail*. Computer Science Department, University of Bristol, U.K.

Tchoukalov, Tzvetan, Christian Monson, and Brian Roark. 2009. Multiple sequence alignment for morphology induction. *Working Notes, CLEF 2009 Workshop*.

Wicentowski, Richard 2002. *Modeling and Learning Multilingual Inflectional Morphology in a Minimally Supervised Framework*. Ph.D. thesis, The Johns Hopkins University, Baltimore, Maryland, U.S.A.

Modeling Socio-Cultural Phenomena in Discourse

Tomek Strzalkowski[1,2], **George Aaron Broadwell**[1], **Jennifer Stromer-Galley**[1], **Samira Shaikh**[1], **Sarah Taylor**[3] **and Nick Webb**[1]
[1]ILS Institute, University at Albany, SUNY
[2]IPI, Polish Academy of Sciences
[3]Lockheed Martin Corporation
tomek@albany.edu

Abstract

In this paper, we describe a novel approach to computational modeling and understanding of social and cultural phenomena in multi-party dialogues. We developed a two-tier approach in which we first detect and classify certain social language uses, including topic control, disagreement, and involvement, that serve as first order models from which presence the higher level social constructs such as leadership, may be inferred.

1. Introduction

We investigate the language dynamics in small group interactions across various settings. Our focus in this paper is on English online chat conversations; however, the models we are developing are more universal and applicable to other conversational situations: informal face-to-face interactions, formal meetings, moderated discussions, as well as interactions conducted in languages other than English, e.g., Urdu and Mandarin.

Multi-party online conversations are particularly interesting because they become a pervasive form of communication within virtual communities, ubiquitous across all age groups. In particular, a great amount of communication online occurs in virtual chat-rooms, typically conducted using a highly informal text dialect. At the same time, the reduced-cue environment of online interaction necessitates more explicit linguistic devices to convey social and cultural nuances than is typical in face-to-face or even voice conversations.

Our objective is to develop computational models of how certain social phenomena such as leadership, power, and conflict are signaled and reflected in language through the choice of lexical, syntactic, semantic and conversational forms by discourse participants. In this paper we report the results of an initial phase of our work during which we constructed a prototype system called DSARMD-1 (Detecting Social Actions and Roles in Multiparty Dialogue). Given a representative segment of multiparty task-oriented dialogue, DSARMD-1 automatically classifies all discourse participants by the degree to which they deploy selected *social language uses*, such as topic control, task control, involvement, and disagreement. These are the mid-level social phenomena, which are deployed by discourse participants in order to achieve or assert higher-level *social constructs*, including leadership. In this work we adopted a two-tier empirical approach where social language uses are modeled through *observable* linguistic features that can be automatically extracted from dialogue. The high-level social constructs are then inferred from a combination of language uses attributed to each discourse participant; for example, a high degree of influence and a high degree of involvement by the same person may indicate a leadership role. In this paper we limit our discussion to the first tier only: how to effectively model and classify social language uses in multi-party dialogue.

2. Related Research

Issues related to linguistic manifestation of social phenomena have not been systematically researched before in computational linguistics; indeed, most of the effort thus far was directed towards the communicative dimension of discourse. While the Speech Acts theory (Austin, 1962; Searle, 1969) provides a generalized framework for multiple levels of discourse analysis (locution, illocution and perlocution), most current approaches to dialogue focus on information content and structural components (Blaylock, 2002; Carberry & Lambert, 1999; Stolcke, et al., 2000) in dialogue; few take into account the effects that speech acts may have upon the social

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1038–1046,
Beijing, August 2010

roles of discourse participants. Also relevant is research on modeling sequences of dialogue acts – to predict the next one (Samuel et al. 1998; Ji & Bilmes, 2006 inter alia) – or to map them onto subsequences or "dialogue games" (Carlson 1983; Levin et al., 1998), which are attempts to formalize participants' roles in conversation (e.g., Linell, 1990; Poesio & Mikheev, 1998; Field et al., 2008).

There is a body of literature in anthropology, linguistics, sociology, and communication on the relationship between language and power, as well as other social phenomena, e.g., conflict, leadership; however, existing approaches typically look at language use in situations where the social relationships are known, rather than using language predictively. For example, conversational analysis (Sacks et al., 1974) is concerned with the structure of interaction: turn-taking, when interruptions occur, how repairs are signaled, but not what they reveal about the speakers. Research in anthropology and communication has concentrated on how certain social norms and behaviors may be reflected in language (e.g., Scollon and Scollon, 2001; Agar, 1994) with few systematic studies attempting to explore the reverse, i.e., what the linguistic phenomena tell us about social norms and behaviors.

3. Data & Annotation

Our initial focus has been on on-line chat dialogues. While chat data is plentiful on-line, its adaptation for research purposes presents a number of challenges that include users' privacy issues on the one hand, and their complete anonymity on the other. Furthermore, most data that may be obtained from public chat-rooms is of limited value for the type of modeling tasks we are interested in due to its high-level of noise, lack of focus, and rapidly shifting, chaotic nature, which makes any longitudinal studies virtually impossible. To derive complex models of conversational behavior, we need the interaction to be reasonably focused on a task and/or social objectives within a group.

Few data collections exist covering multiparty dialogue, and even fewer with on-line chat. Moreover, the few collections that exist were built primarily for the purpose of training dialogue act tagging and similar linguistic phenomena; few if any of these corpora are suitable for deriving pragmatic models of conversation, including socio-linguistic phenomena. Existing resources include a multi-person meeting corpus ICSI-MRDA and the AMI Meeting Corpus (Carletta, 2007), which contains 100 hours of meetings captured using synchronized recording devices. Still, all of these resources look at spoken language rather than on-line chat. There is a parallel interest in the online chat environment, although the development of useful resources has progressed less. Some corpora exist such as the NPS Internet chat corpus (Forsyth and Martell, 2007), which has been hand-anonymized and labeled with part-of-speech tags and dialogue act labels. The StrikeCom corpus (Twitchell et al., 2007) consists of 32 multi-person chat dialogues between players of a strategic game, where in 50% of the dialogues one participant has been asked to behave 'deceptively'.

It is thus more typical that those interested in the study of Internet chat compile their own corpus on an as needed basis, e.g., Wu et al. (2002), Khan et al. (2002), Kim et al. (2007).

Driven by the need to obtain a suitable dataset we designed a series of experiments in which recruited subjects were invited to participate in a series of on-line chat sessions in a specially designed secure chat-room. The experiments were carefully designed around topics, tasks, and games for the participants to engage in so that appropriate types of behavior, e.g., disagreement, power play, persuasion, etc. may emerge spontaneously. These experiments and the resulting corpus have been described elsewhere (Shaikh et al., 2010b), and we refer the reader to this source. Ultimately a corpus of 50 hours of English chat dialogue was collected comprising more than 20,000 turns and 120,000 words. In addition we also assembled a corpus of 20 hours of Urdu chat.

A subset of English language dataset has been annotated at four levels: communication links, dialogue acts, local topics and meso-topics (which are essentially the most persistent local topics). Although full details of these annotations are impossible to explain within the scope of this article, we briefly describe them below. Annotated datasets were used to develop and train automatic modules that detect and classify social uses of language in discourse. It is important to note that the annota-

tion has been developed to support the objectives of our project and does not necessarily conform to other similar annotation systems used in the past.

- *Communicative links*. In a multi-party dialogue an utterance may be directed towards a specific participant, a subgroup of participants or to everyone.
- *Dialogue Acts*. We developed a hierarchy of 15 dialogue acts for annotating the functional aspect of the utterance in discussion. The tagset we adopted is based on DAMSL (Allen & Core, 1997) and SWBD (Jurafsky et al., 1997), but compressed to 15 tags tuned significantly towards dialogue pragmatics and away from more surface characteristics of utterances (Shaikh et al., 2010a).
- *Local topics*. Local topics are defined as nouns or noun phrases introduced into discourse that are subsequently mentioned again via repetition, synonym, or pronoun.
- *Topic reference polarity*. Some topics, which we call *meso-topics*, persist through a number of turns in conversation. A selection of meso-topics is closely associated with the task in which the discourse participants are engaged. Meso-topics can be distinguished from the local topics because the speakers often make polarized statements about them.

4. Socio-linguistic Phenomena

We are interested in modeling the social phenomena of Leadership and Power in discourse. These high-level phenomena (or Social Roles, *SR*) will be detected and attributed to discourse participants based on their deployment of selected *Language Uses* (LU) in multi-party dialogue. Language Uses are mid-level socio-linguistic devices that link linguistic components deployed in discourse (from lexical to pragmatic) to social constructs obtaining for and between the participants. The language uses that we are currently studying are *Agenda Control*, *Disagreement*, and *Involvement* (Broadwell et al., 2010).

Our research so far is focused on the analysis of English-language synchronous chat, and we are looking for correlations between various metrics that can be used to detect LU in multiparty dialogue. We expect that some of these correlations may be culturally specific or language-specific, as we move into the analysis of Urdu and Mandarin discourse in the next phase of this project.

4.1 Agenda Control in Dialogue

Agenda Control is defined as efforts by a member or members of the group to advance the group's task or goal. This is a complex LU that we will model along two dimensions: (1) *Topic Control* and (2) *Task Control*. Topic Control refers to attempts by any discourse participants to impose the topic of conversation. Task Control, on the other hand, is an effort by some members of the group to define the group's project or goal and/or steer the group towards that goal. We believe that both behaviors can be detected using scalar measures per participant based on certain linguistic features of their utterances.

For example, one hypothesis is that topic control is indicated by the rate of *local topic introductions* (LTI) per participant (Givon, 1983). Local topics may be defined quite simply as noun phrases introduced into discourse, which are subsequently mentioned again via repetition, synonym, pronoun, or other form of co-reference. Thus, one measure of topic control is the number of local topics introduced by each participant as percentage of all local topics in a discourse.

Using an LTI index we can construct assertions about topic control in a discourse. For example, suppose the following information is discovered about the speaker LE in a multi-party discussion *dialogue-1*[1] where 90 local topics are identified:

1. LE introduces 23/90 (25.6%) of local topics in this dialogue.
2. The mean rate of local topic introductions is this dialogue is 14.29%, and standard deviation is 8.01.
3. LE is in the top quintile of participants for introducing new local topics

We can now claim the following, with a degree of confidence (to be determined):

$$TopicControl_{LTI}(LE, 5, dialogue\text{-}1)$$

We read this as follows: *speaker LE exerts the highest degree of topic control in dialogue-1*. Of course, LTI is just one source of evidence and we developed other metrics to complement it. We mention three of them here:

[1] Dialogue-1 refers to an actual dataset of 90-minute chat among 7 participants, covering approximately 700 turns. The task is to select a candidate for a job given a set of resumes.

- *SMT Index*. This is a measure of topic control suggested in (Givon, 1983) and it is based on subsequent mentions of already introduced local topics. Speakers who introduce topics that are discussed at length by the group tend to control the topic of the discussion. The *subsequent mentions of local topics* (SMT) index calculates the percentage of second and subsequent references to the local topics, by repetition, synonym, or pronoun, relative to the speakers who introduced them.
- *Cite Score*. This index measures the extent to which other participants discuss topics introduced by that speaker. The difference between SMT and CiteScore is that the latter reflect to what degree a speaker's efforts to control the topic are assented to by other participants in a conversation.
- *TL Index* (TL). This index stipulates that more influential speakers take longer turns than those who are less influential. The TL index is defined as the average number of words per turn for each speaker. Turn length also reflects the extent to which other participants are willing to 'yield the floor' in conversation.

Like LTI, all the above indices are mapped into a degree of topic control, based on quintiles in normal distribution (Table 1).

	LTI	**SMT**	**CS**	**TL**	AVG
LE	5	5	5	5	5.00
JR	4	4	4	3	3.75
KI	4	3	3	1	2.75
KN	3	5	4	4	4.00
KA	2	2	2	4	2.50
CS	2	2	2	2	2.00
JY	1	1	1	2	1.25

Table 1: Topic Control distribution in dialogue-1. Each row represents a speaker in the group (LE, JR, etc.). Columns show indices used, with degrees per speaker on 5-point scale based on quintiles in normal distribution, and the average value.

Ideally, all the above indices (and others yet to be defined) should predict the same outcome, i.e., for each dialogue participant they should assign the same degree of topic control, relative to other speakers. This is not always the case, and where the indices divert in their predictions, our level of confidence in the generated claims decreases. We are currently working on how these different metrics correlate to each other and how they should be weighted to maximize accuracy of making Topic Control claims. Nonetheless, we can already output a Topic Control map (shown in Table 1) that captures a sense of internal social dynamics within the group.

The other aspect of Agenda Control phenomenon is Task Control. It is defined as an effort to determine the group's goal and/or steer the group towards that goal. Unlike Topic Control, which is imposed by influencing the subject of conversation, Task Control is gained by directing other participants to perform certain tasks or accept certain opinions. Consequently, Task Control is detected by observing the usage of certain dialogue acts, including Action-Directive, Agree-Accept, Disagree-Reject, and related categories. Here again, we define several indices that allow us to compute a degree of Task Control in dialogue for each participant:

- *Directive Index* (DI). The participant who directs others is attempting to control the course of the task that the group is performing. We count the number of directives, i.e., utterances classified as Action-Directive, made by each participant as a percentage of all directives in discourse.
- *Directed Topic Shift Index* (DTSI). When a participant who controls the task offers a directive on the task, then the topic of conversation shifts. In order to detect this condition, we calculate the ratio of coincidence of directive dialogue acts by each participant with topic shifts following them.
- *Process Management index* (PMI). Another measure of Task Control is the proportion of turns each participant has that explicitly address the problem solving process. This includes utterances that involve coordinating the activities of the participants, planning the order of activities, etc. These fall into the category of Task (or Process) Management in most DA tagging systems.
- *Process Management Success Index* (PMSI). This index measures the degree of success by each speaker at controlling the task. A credit is given to the speaker whose suggested curse of action is supported by other speakers for each response that supports the suggestion. Conversely, a credit is taken away for each response that rejects or

qualifies the suggestion. PMSI is computed as distribution of task management credits among the participants over all dialogue utterances classified as Task/Process Management. [2]

As an example, let's consider the following information computed for the PMI index over *dialogue-1*:

1. *Dialogue-1* contains 246 utterances classified as Task/Process Management rather than doing the task.
2. Speaker KI makes 65 of these utterances for a PMI of 26.4%.
3. Mean PMI for participants is 14.3%; 80th percentile is >21.2%. PMI for KI is in the top quintile for all participants.

Based on this evidence we may claim (with yet to be determined confidence) that:

$$TaskControl_{PMI}(KI, 5, dialogue\text{-}1)$$

This may be read as follows: *speaker KI exerts the highest degree of Task Control in dialogue-1*. We note that Task Control and Topic Control do not coincide in this discourse, at least based on the PMI index. Other index values for Task Control may be computed and tabulated in a way similar to LTI in Table 1. We omit these here due to space limitations.

4.2 Disagreement in Dialogue

Disagreement is another language use that correlates with speaker's power and leadership. There are two ways in which disagreement is realized: *expressive disagreement* and *topical disagreement* (Stromer-Galley, 2007; Price, 2002). Both can be detected using scalar measures applied to subsets of participants, typically any two participants. In addition, we can also measure for each participant the rate with which he or she generates disagreement (with any and all other speakers). *Expressive Disagreement* is normally understood at the level of dialogue acts, i.e., when discourse participants make explicit utterances of disagreement, disapproval, or rejection in response to a prior speaker's utterance. Here is an example (KI and KA are two speakers in a multiparty dialogue in which participants discuss candidates for a youth counselor job):

KA: *CARLA... women are always better with kids*
KI: *That's not true!*
KI: *Men can be good with kids too*

While such exchanges are vivid examples of expressive disagreement, we are interested in more sustained phenomenon where two speakers repeatedly disagree, thus revealing a social relationship between them. Therefore, one measure of Expressive Disagreement that we consider is the number of Disagree-Reject dialogue acts between any two speakers as a percentage of all utterances exchanged between these two speakers. This becomes a basis for the *Disagree-Reject Index* (DRX). In dialogue-1 we have:

1. Speakers KI and KA have 47 turns between them. Among these there are 8 turns classified as Disagree-Reject, for the DRX of 15.7%.
2. The mean DRX for speakers who make any Disagree-Reject utterances is 9.5%. The pair of speakers KI-KA is in the top quintile (>13.6%).

Based on this evidence we can conclude the following:

$$ExpDisagreement_{DRX}(KI,KA, 5, dialogue\text{-}1)$$

which may be read as follows: *speakers KI and KA have the highest level of expressive disagreement in dialogue-1*. This measure is complemented by a *Cumulative Disagreement Index* (CDX), which is computed for each speaker as a percentage of all Disagree-Reject utterances in the discourse that are made by this speaker. Unlike DRX, which is computed for pairs of speakers, the CDX values are assigned to each group participant and indicate the degree of disagreement that each person generates.

While Expressive Disagreement is based on the use of more overt linguistic devices, *Topical Disagreement* is defined as a difference in referential *valence* in utterances (statements, opinions, questions, etc.) made on a topic. Referential valence of an utterance is determined by the type of statement made about the topic in question, which can be positive (+), negative (–), or neutral (0). A positive statement is one in favor of (*express advocacy*) or in support of (*supporting information*) the topic being discussed. A negative statement is one that is against or negative on

[2] The exact structure of the credit function is still being determined experimentally. For example, more credit may be given to first supporting response and less for subsequent responses; more credit may be given for unprompted suggestions than for those that were responding to questions from others.

the topic being discussed. A neutral statement is one that does not indicate the speaker's position on the topic. Here is an example of opposing polarity statements about the same topic in discourse:

Sp-1: *I like that he mentions "Volunteerism and Leadership"*

Sp-2: *but if they're looking for someone who is experienced then I'd cross him off*

Detecting topical disagreement in discourse is more complicated because its strength may vary from one topic in a conversation to the next. A reasonable approach is thus to measure the degree of disagreement between two speakers on one topic first, and then extrapolate over the entire discourse. Accordingly, our measure of topical disagreement is valuation differential between any two speakers as expressed in their utterances about a topic. Here, the topic (or an "issue") is understood more narrowly than the local topic defined in the previous section (as used in Topic Control, for example), and may be assumed to cover only the most persistent local topics, i.e., topics with the largest number of references in dialogue, or what we call the *meso-topics*. For example, in a discussion of job applicants, each of the applicants becomes a meso-topic, and there may be additional meso-topics present, such as qualifications required, etc.

The resulting *Topical Disagreement Metric* (TDM) captures the degree to which any two speakers advocate the opposite sides of a meso-topic. TDM is computed as an average of *P*-valuation differential for one speaker (advocating *for* a meso-topic) and (*–P*)-valuation differential for the other speaker (advocating *against* the meso-topic).

Using TDM we can construct claims related to disagreement in a given multiparty dialogue of sufficient duration (exactly what constitutes a sufficient duration is still being researched). Below is an example based on a 90-minute chat dialogue-1 about several job candidates for a youth counselor. The discussion involved 7 participants, including KI and KA. Topical disagreement is measured on 5 points scale (corresponding to quintiles in normal distribution):

$TpDisAgree_{TDM}$(*KI,KA,"Carla",4,dialogue-1*)

This may be read as follows: speakers KI and KA *topically disagree to degree 4* on topic [job candidate] "Carla" in dialogue-1. In order to calculate this we compute the value of TDM index between these two speakers. We find that KA makes 30% of all positive utterances made by anyone about Carla (40), while KI makes 45% of all negative utterances against Carla. This places these two speakers in the top quintiles in the "for Carla" polarity distribution and "against Carla" distribution, respectively. Taking into account any opposing polarity statements made by KA against Carla and any statements made by KI for Carla, we calculate the level of topical disagreement between KA and KI to be 4 on the 1-5 scale.

TDM allows us to compute topical disagreement between any two speakers in a discourse, which may also be represented in a 2-dimensional table revealing another interesting aspect of internal group dynamics.

4.3 Involvement in Dialogue

The third type of social language use that we discuss in this paper is Involvement. Involvement is defined as a degree of engagement or participation in the discussion of a group. It is an important element of leadership, although its importance is expected to differ between cultures; in Western cultures, high involvement and influence (topic control) often correlates with group leadership.

In order to measure Involvement we designed several indices based on turn characteristics for each speaker. Four of the indices are briefly explained below:

- The *NP index* (NPI) is a measure of gross informational content contributed by each speaker in discourse. *NPI* counts the ratio of third-person nouns and pronouns used by a speaker to the total number of nouns and pronouns in the discourse.
- The *Turn index* (*TI*) is a measure of *interactional frequency*; it counts the ratio of turns per participant to the total number of turns in the discourse.
- The *Topic Chain Index* (*TCI*) counts the degree to which participants discuss of the most persistent topics. In order to calculate TCI values, we define a *topic chains* for all local topics. We compute frequency of mentions of these longest topics for each participant.
- The *Allotopicality Index* (*ATP*) counts the number of mentions of local topics that were introduced by other participants. An

ATP value is the proportion of a speaker's allotopical mentions, i.e., excluding "self-citations", to all allotopical mentions in a discourse.

As an example, we may consider the following situation in *dialogue-1*:

1. *Dialogue-1* contains 796 third person nouns and pronouns, excluding mentions of participants' names.
2. Speaker JR uses 180 nouns and pronouns for an NPI of 22.6%.
3. The median NPI is 14.3%; JR are in the upper quintile of participants (> 19.9%).

From the above evidence we can draw the following claim:

$$Involvement_{NPI}(JR, 5, dialogue\text{-}1)$$

This may be read as: *speaker JR is the most involved participant in dialogue-1.*

As with other language uses, multiple indices for Involvement can be combined into a 2-dimensional map capturing the group internal dynamics.

5. Implementation & Evaluation

We developed a prototype automated DSARMD system that comprises a series of modules that create automated annotation of the source dialogue for all the language elements discussed above, including communicative links, dialogue acts, local/meso topics, and polarity. Automatically annotated dialogue is then used to generate language use degree claims. In order to evaluate accuracy of the automated process we conducted a preliminary evaluation comparing the LU claims generated from automatically annotated data to the claims generated from manually coded dialogues. Below we briefly describe the methodology and metrics used.

Each language use is asserted per a participant in a discourse (or per each pair of participants, e.g., for Disagreement) on a 5-point "strength" scale. This can be represented as an ordered sequence $LU_X(d_1, d_2, \ldots d_n)$, where LU is the language use being asserted, X is the index used, d_i is the degree of LU attributed to speaker i. This assignment is therefore a 5-way classification of all discourse participants and its correctness is measured by dividing the number of correct assignments by the total number of elements to be classified, which gives the micro-averaged precision. The accuracy metric is computed with several variants as follows:

1. *Strict mapping*: each complete match is counted as 1; all mismatches are counted as 0. For example, the outputs LU_X (5,4,3,2,1) and LU_X (4,5,3,1,1) produce two exact matches (for the third and the last speaker) for a precision of 0.4.
2. *Weighted mapping*: since each degree value d_i in $LU_X(d_1, d_2, \ldots d_n)$ represents a quintile in normal distribution, we consider the position of the value within the quintile. If two mismatched values are less than ½ quintile apart we assign a partial credit (currently 0.5).
3. *Highest – Rest*: we measure accuracy with which the highest LU degree (but not necessarily the same degree) is assigned to the right speaker vs. any other score. This results in binary classification of scores. The sequences in (1) produce 0.6 match score.
4. *High – Low*: An alternative binary classification where scores 5 and 4 are considered High, while the remaining scores are considered Low. Under this metric, the sequences in (1) match with 100% precision.

The process of automatic assignment of language uses derived from automatically processed dialogues was evaluated against the control set of assignments based on human-annotated data. In order to obtain a reliable "ground truth", each test dialogue was annotated by at least three human coders (linguistics and communication graduate students, trained). Since human annotation was done at the linguistic component level, a strict inter-annotator agreement was not required; instead, we were interested whether in each case a comparable statistical distribution of the corresponding LU index was obtained. Annotations that produced index distributions dissimilar from the majority were eliminated.

Automated dialogue processing involved the following modules:

- *Local topics detection* identifies first mentions by tracking occurrences of noun phrases. Subsequent mentions are identified using fairly simple pronoun resolution (based mostly on lexical features), with Wordnet used to identify synonyms, etc.
- *Meso-topics* are identified as longest-chain local topics. Their *polarity* is assessed at the utterance level by noting presence of positive or negative cue words and phrases.
- *Dialogue acts* are tagged based on presence

of certain cue phrases derived from a training corpus (Webb et al., 2008).
- *Communicative links* are mapped by computing inter-utterance similarity based on n-gram overlap.

Preliminary evaluation results are shown in Tables 3-5 with average performance over 3 chat sessions (approx 4.5 hours) involving three groups of speakers and different tasks (job candidates, political issues). Topic Control and Involvement tables show average accuracy per index. For example, the LTI index, computed over automatically extracted local topics, produces Topic Control assignments with the average precision of 80% when compared to assignments derived from human-annotated data using the strict accuracy metric. However, automated prediction of Involvement based on NPI index is far less reliable, although we can still pick the most involved speaker with 67% accuracy. We omit the indices based on turn length (TL) and turn count (TI) because their values are trivially computed. At this time we do not combine indices into a single LU prediction. Additional experiments are needed to determine how much each of these indices contributes to LU prediction.

Topic Control	LTI	SMT	CS
Strict	0.80	0.40	0.40
Weighted	0.90	0.53	0.53
Highest-Rest	0.90	0.67	0.67
High-Low	1.00	0.84	0.90

Table 3: Topic Control LU assignment performance averages of selected indices over a subset of data covering three dialogues with combined duration of 4.5 hours with total of 19 participants (7, 5, 7 per session).

Involvement	NPI	TCI	ATP
Strict	0.31	0.42	0.39
Weighted	0.46	0.49	0.42
Highest-Rest	0.67	0.77	0.68
High-Low	0.58	0.74	0.48

Table 4: Involvement LU assignment performance averages for selected indices over the same subset of data as in Table 3.

Topical Disagreement performance is shown in Table 5. We calculated precision and recall of assigning a correct degree of disagreement to each pair of speakers who are members of a group. Precision and recall averages are then computed over all meso-topics identified in the test dataset, which consists of three separate 90-minute dialogues involving 7, 5 and 7 speakers, respectively. Our calculation includes the cases where different sets of meso-topics were identified by the system and by the human coder. A strict mapping of levels of disagreement between speakers is hard to compute accurately; however, finding the speakers who disagree the most, or the least, is significantly more robust.

Topical Disagreement	Prec.	Recall
Strict	0.33	0.32
Weighted	0.54	0.54
Highest-Rest	0.89	0.85
High-Low	0.77	0.73

Table 5: Topical Disagreement LU assignment performance averages over 13 meso-topics discussed in three dialogues with combined duration of 4.5 hours with total of 19 participants (7, 5, and 7 per session).

6. Conclusion

In this paper we presented a preliminary design for modeling certain types of social phenomena in multi-party on-line dialogues. Initial, limited-scale evaluation indicates that the model can be effectively automated. Much work lies ahead, including large scale evaluation, testing index stability and resilience to NL component level error. Current performance of the system is based on only preliminary versions of linguistic modules (topic extraction, polarity assignments, etc.) which perform at only 70-80% accuracy, so these need to be improved as well. Research on Urdu and Chinese dialogues is just starting.

Acknowledgements

This research was funded by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), through the U.S. Army Research Lab. All statements of fact, opinion or conclusions contained herein are those of the authors and should not be construed as representing the official views or policies of IARPA, the ODNI or the U.S. Government.

References

Agar, Michael. 1994. Language Shock, Understanding the Culture of Conversation. Quill, William Morrow, New York.

Allen, J. M. Core. 1997. Draft of DAMSL: Dialog Act Markup in Several Layers. www.cs. rochester.edu/research/cisd/resources/damsl/

Anderson, A., et al. 1991. The HCRC Map Task Corpus. Language and Speech 34(4), 351--366.

Austin, J. L. 1962. *How to do Things with Words*. Clarendon Press, Oxford.

Bird, Steven, et al. 2009. Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. O'Reilly Media.

Blaylock, Nate. 2002. Managing Communicative Intentions in Dialogue Using a Collaborative Problem-Solving Model. Technical Report 774, University of Rochester, CS Dept.

Broadwell, G. A et al. (2010). Social Phenomena and Language Use. ILS Technical report.

Carberry, Sandra and Lynn Lambert. 1999. A Process Model for Recognizing Communicative Acts and Modeling Negotiation Dialogue. *Computational Linguistics*, 25(1), pp. 1-53.

Carletta, J. (2007). Unleashing the killer corpus: experiences in creating the multi-everything AMI Meeting Corpus. Language Resources and Evaluation Journal 41(2): 181-190

Carlson, Lauri. 1983. *Dialogue Games: An Approach to Discourse Analysis*. D. Reidel.

Eric N. Forsyth and Craig H. Martell. 2007. Lexical and Discourse Analysis of Online Chat Dialog. First IEEE International Conference on Semantic Computing (ICSC 2007), pp. 19-26.

Field, D., et al. 2008. Automatic Induction of Dialogue Structure from the Companions Dialogue Corpus, 4th Int. Workshop on Human-Computer Conversation, Bellagio.

Givon, Talmy. 1983. Topic continuity in discourse: A quantitative cross-language study. Amsterdam: John Benjamins.

Ivanovic, Edward. 2005. Dialogue Act Tagging for Instant Messaging Chat Sessions. In Proceedings of the ACL Student Research Workshop. 79–84. Ann Arbor, Michigan.

Ji, Gang Jeff Bilmes. 2006. Backoff Model Training using Partially Observed Data: Application to Dialog Act Tagging. HLT-NAACL

Jurafsky, Dan, Elizabeth Shriberg, and Debra Biasca. 1997. Switchboard SWBD-DAMSL Shallow-Discourse-Function Annotation Coders Manual. http://stripe.colorado.edu/~jurafsky/manual.august1.html

Jurafsky, D., et al. 1997. Automatic detection of discourse structure for speech recognition and understanding. IEEE Workshop on Speech Recognition and Understanding, Santa Barbara.

Khan, Faisal M., et al. 2002. Mining Chat-room Conversations for Social and Semantic Interactions. Computer Science and Engineering, Lehigh University.

Kim, Jihie., et al. 2007. An Intelligent Discussion-Bot for Guiding Student Interactions in Threaded Discussions. AAAI Spring Symposium on Interaction Challenges for Intelligent Assistants

Levin, L., et al. (1998). A discourse coding scheme for conversational Spanish. International Conference on Speech and Language Processing.

Levin, L., et al. (2003). Domain specific speech acts for spoken language translation. 4th SIGdial Workshop on Discourse and Dialogue.

Linell, Per. 1990. The power of dialogue dynamics. In Ivana Markov´a and Klaus Foppa, editors, *The Dynamics of Dialogue*. Harvester, 147–177.

Poesio, Massimo and Andrei Mikheev. 1998. The predictive power of game structure in dialogue act recognition. International Conference on Speech and Language Processing *(ICSLP-98)*.

Price, V., Capella, J. N., & Nir, L. (2002). Does disagreement contribute to more deliberative opinion? Political Communication, 19, 95-112.

Sacks, H. and Schegloff, E., Jefferson, G. 1974. A simplest systematic for the organization of turn-taking for conversation. In: *Language* 50(4), 696-735.

Samuel, K. et al. 1998. Dialogue Act Tagging with Transformation-Based Learning. 36th Annual Meeting of the ACL.

Scollon, Ron and Suzanne W. Scollon. 2001. *Intercultural Communication, A Discourse Approach*. Blackwell Publishing, Second Edition.

Searle, J. R. 1969. *Speech Acts*. Cambridge University Press, London-New York.

Shaikh, S. et al. 2010. DSARMD Annotation Guidelines, V. 2.5. ILS Technical Report.

Shaikh S. et al. 2010. MPC: A Multi-Party Chat Corpus for Modeling Social Phenomena in Discourse, Proc. LREC-2010, Malta.

Stolcke, Andreas et al. 2000. Dialogue Act Modeling for Automatic Tagging and Recognition of Conversational Speech. *Computational Linguistics*, 26(3).

Stromer-Galley, J. 2007. Measuring deliberation's content: A coding scheme. Journal of Public Deliberation, 3(1).

Tianhao Wu, et al. 2002. Posting Act Tagging Using Transformation-Based Learning. Foundations of Data Mining and Discovery, IEEE International Conference on Data Mining

Twitchell, Douglas P., Jay F. Nunamaker Jr., and Judee K. Burgoon. 2004. Using Speech Act Profiling for Deception Detection. Intelligence and Security Informatics, LNCS, Vol. 3073

Webb, N., T. Liu, M. Hepple and Y. Wilks. 2008. Cross-Domain Dialogue Act Tagging. 6th International Conference on Language Resources and Evaluation (LREC-2008), Marrakech.

Discriminative Induction of Sub-Tree Alignment using Limited Labeled Data

Jun Sun[1,2] **Min Zhang**[1] **Chew Lim Tan**[2]
[1]Institute for Infocomm Research [2]School of Computing, National University of Singapore
sunjun@comp.nus.edu.sg mzhang@i2r.a-star.edu.sg tancl@comp.nus.edu.sg

Abstract

We employ Maximum Entropy model to conduct sub-tree alignment between bilingual phrasal structure trees. Various lexical and structural knowledge is explored to measure the syntactic similarity across Chinese-English bilingual tree pairs. In the experiment, we evaluate the sub-tree alignment using both gold standard tree bank and the automatically parsed corpus with manually annotated sub-tree alignment. Compared with a heuristic similarity based method, the proposed method significantly improves the performance with only limited sub-tree aligned data. To examine its effectiveness for multilingual applications, we further attempt different approaches to apply the sub-tree alignment in both phrase and syntax based SMT systems. We then compare the performance with that of the widely used word alignment. Experimental results on benchmark data show that sub-tree alignment benefits both systems by relaxing the constraint of the word alignment.

1 Introduction

Recent research in Statistical Machine Translation (SMT) tends to incorporate more linguistically grammatical information into the translation model known as linguistically motivated syntax-based models. To develop such models, the phrasal structure parse tree is usually adopted as the representation of bilingual sentence pairs either on the source side (Huang et al., 2006; Liu et al., 2006) or on the target side (Galley et al., 2006; Marcu et al., 2006), or even on both sides (Graehl and Knight, 2004; Zhang et al., 2007). Most of the above models either construct a pipeline to transform from/to tree structure, or synchronously generate two trees in parallel (i.e., synchronous parsing). Both cases require syntactically rich translational equivalences to handle non-local reordering. However, most current works obtain the syntactic translational equivalences by initially conducting alignment on the word level. To employ word alignment as a hard constraint for rule extraction has difficulty in capturing such non-local phenomena and will fully propagate the word alignment error to the later stage of rule extraction.

Alternatively, some initial attempts have been made to directly conduct syntactic structure alignment. As mentioned in Tinsley et al. (2007), the early work usually constructs the structure alignment by hand, which is time-consuming. Recent research tries to automatically align the bilingual syntactic sub-trees. However, most of these works suffer from the following problems. Firstly, the alignment is conducted based on heuristic rules, which may lose extensibility and generality in spite of accommodating some common cases (Groves et al., 2004). Secondly, various similarity computation methods are used based merely on lexical translation probabilities (Tinsley et al., 2007; Imamura, 2001) regardless of structural features. We believe the structure information is an important issue to capture the non-local structural divergence of languages by modeling beyond the plain text.

To address the above issues, we present a statistical framework based on Maximum Entropy (MaxEnt) model. Specifically, we consider sub-tree alignment as a binary classification problem and use Maximum Entropy model to classify each instance as *aligned* or *unaligned*. Then, we perform a greedy search within the reduced search space to conduct sub-tree alignment links based on the alignment probabilities obtained from the classifier.

Unlike the previous approaches that can only measure the structural divergence via lexical features, our approach can incorporate both lexical and structural features. Additionally, instead of explicitly describing the instances of sub-tree pairs as factorized sub-structures, we frame most of our features as score based feature functions, which helps solve the problem using limited sub-tree alignment annotated data. To train the model and evaluate the alignment performance, we adopt

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1047–1055,
Beijing, August 2010

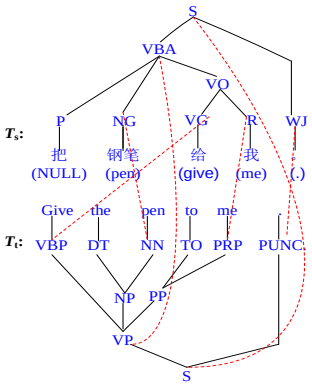


Figure 1: Sub-tree alignment as referred to Node alignment

HIT Chinese-English parallel tree bank for gold standard evaluation. To explore its effectiveness in SMT systems, we also manually annotate sub-tree alignment on automatically parsed tree pairs and perform the noisy data evaluation. Experimental results show that by only using limited sub-tree aligned data of both corpora, the proposed approach significantly outperforms the baseline method (Tinsley et al., 2007). The proposed features are very effective in modeling the bilingual structural similarity. We further apply the sub-tree alignment to relax the constraint of word alignment for both phrase and syntax based SMT systems and gain an improvement in BLEU.

2 Problem definition

A sub-tree alignment process pairs up the sub-trees across bilingual parse trees, whose lexical leaf nodes covered are translational equivalent, i.e., sharing the same semantics. Grammatically, the task conducts links between syntactic constituents with the maximum tree structures generated over their word sequences in bilingual tree pairs.

In general, sub-tree alignment can also be interpreted as conducting multiple links across internal nodes between sentence-aligned tree pairs as shown in Fig. 1. The aligned sub-tree pairs usually maintain a non-isomorphic relation with each other especially for higher layers. We adapt the same criteria as Tinsley et al. (2007) in our study:

(i) a node can only be linked once;
(ii) descendants of a source linked node may only link to descendants of its target linked counterpart;
(iii) ancestors of a source linked node may only link to ancestors of its target linked counterpart.

where the term "node" refers to root of a sub-tree, which can be used to represent the sub-tree.

3 Model

We solve the problem as binary classification and employ MaxEnt model with a greedy search.

Given a bilingual tree pair S^I and T^J, $S^I = \{s_1, \dots, s_i, \dots, s_I\}$ is the source tree consisting of I sub-trees, where I is also the number of nodes in the source tree S^I. $T^J = \{t_1, \dots, t_j, \dots, t_J\}$ is the target tree consisting of J sub-trees, where J is also the number of nodes in the target tree T^J.

For each sub-tree pair (s_i, t_j) in the given bilingual parse trees (S^I, T^J), the sub-tree alignment probability is given by:

$$Pr(a|s_i, t_j, \theta) = \frac{\exp[\sum_{m=1}^{M} \lambda_m h_m(a, s_i, t_j, \theta)]}{\sum_{a'} \exp[\sum_{m=1}^{M} \lambda_m h_m(a', s_i, t_j, \theta)]} \quad (1)$$

where

$$a = \begin{cases} 1 & \text{if } (s_i, t_j) \text{ is aligned} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Feature functions are defined in a quadruple (a, s_i, t_j, θ). θ is an additional variable to incorporate new dependencies other than the sub-tree pairs. For each feature function $h_m(a, s_i, t_j, \theta)$, a weight λ_m is applied to tailor the distribution.

After classifying the candidate sub-tree pairs as *aligned* or *unaligned*, we perform a greedy search within the reduced search space to conduct *sure* links based on the conditional probability $Pr(a|s_i, t_j, \theta)$ obtained from the classifier. The alignment probability is independently normalized for each sub-tree pair and hence suitable as a searching metric.

The greedy search algorithm can be described as an automaton. A state in the search space is a partial alignment with respect to the given bilingual tree pair. A transition is to add one more link of node pairs to the current state. The initial state has no link. The terminal state is a state where no more links can be added according to the definition in Section 2. We use greedy search to generate the best-links at the early stage. There are cases that the correctly-aligned tree pairs have very few links, while we have a bunch of candidates with lower alignment probabilities. However, the sum of the lower probabilities is larger than that of the correct links', since the number of correct links is much fewer. This makes the alignment results biased to be with more links. The greedy search helps avoid this asymmetric problem.

4 Feature Functions

In this section, we introduce a variety of feature functions to capture the semantically equivalent

counterparts and structural divergence across languages. For the semantic equivalence, we define lexical and word alignment feature functions. Since those feature functions are directional, we describe most of these functions as conditional feature functions based on the conditional lexical probabilities. We also introduce the tree structural features to deal with the structural divergence of bilingual parse trees. Inspired by Burkett and Klein (2008), we introduce the feature functions in an internal-external manner based on the fact that the feature scores for an aligned sub-tree pair tend to be high inside both sub-trees, while they tend to be low inside one sub-tree and outside the other.

4.1 Internal Lexical Features

We use this feature to measure the degree of semantic equivalence of the sub-tree pair. According to the definition of sub-tree alignment in Section 2, the word sequence covered by the sub-tree pair should be translational equivalence. Therefore, the lexicons within the two corresponding sub-spans should be highly related in semantics. We define the internal lexical features as follows:

$$\phi(s_i|t_j) = \left(\prod_{v\in in(t_j)} \sum_{u\in in(s_i)} P(u|v)\right)^{\frac{1}{|in(t_j)|}}$$

$$\phi(t_j|s_i) = \left(\prod_{u\in in(s_i)} \sum_{v\in in(t_j)} P(v|u)\right)^{\frac{1}{|in(s_i)|}}$$

where $P(v|u)$ refers to the lexical translation probability from the source word u to the target word v within the sub-tree spans, while $P(u|v)$ refers to that from target to source; $in(s_i)$ refers to the word set for the internal span of the source sub-tree s_i, while $in(t_j)$ refers to that of the target sub-tree t_j.

4.2 Internal-External Lexical Features

Intuitively, lexical translation probabilities tend to be high within the translational equivalence, while low within the non-equivalent counterparts. According to this, we define the internal-external lexical feature functions as follows:

$$\varphi(s_i|t_j) = \frac{\sum_{v\in in(t_j)} \max_{u\in out(s_i)} \left\{(P(u|v)\cdot P(v|u))^{\frac{1}{2}}\right\}}{|in(t_j)|}$$

$$\varphi(t_j|s_i) = \frac{\sum_{u\in in(s_i)} \max_{v\in out(t_j)} \left\{(P(u|v)\cdot P(v|u))^{\frac{1}{2}}\right\}}{|in(s_i)|}$$

where $out(s_i)$ refers to the word set for the external span of the source sub-tree s_i, while $out(t_j)$ refers to that of the target sub-tree t_j. We choose a representation different from the internal lexical feature scores, since for cases with small inner span and large outer span, the sum of internal-external scores may be overestimated. As a result, we change the *sum* operation into *max*, which is easy to be normalized.

4.3 Internal Word Alignment Features

Although the word alignment information within bilingual sentence pairs is to some extent not reliable, the links of word alignment account much for the co-occurrence of the aligned terms. We define the internal word alignment features as follows:

$$\rho(s_i, t_j) = \frac{\sum_{v\in in(t_j)} \sum_{u\in in(s_i)} \delta(u,v)\cdot (P(u|v)\cdot P(v|u))^{\frac{1}{2}}}{(|in(s_i)|\cdot|in(t_j)|)^{\frac{1}{2}}}$$

where

$$\delta(u,v) = \begin{cases} 1 & \text{if } (u,v) \text{ is aligned} \\ 0 & \text{otherwise} \end{cases}$$

The binary function $\delta(u,v)$ is introduced to trigger the computation only when a word aligned link exists for the two words (u,v) within the sub-tree span.

4.4 Internal-External Word Alignment Features

Similar to lexical features, we also introduce internal-external word alignment features as follows:

$$\sigma(s_i|t_j) = \frac{\sum_{v\in in(t_j)} \sum_{u\in out(s_i)} \delta(u,v)\cdot (P(u|v)\cdot P(v|u))^{\frac{1}{2}}}{(|out(s_i)|\cdot|in(t_j)|)^{\frac{1}{2}}}$$

$$\sigma(t_j|s_i) = \frac{\sum_{v\in out(t_j)} \sum_{u\in in(s_i)} \delta(u,v)\cdot (P(u|v)\cdot P(v|u))^{\frac{1}{2}}}{(|in(s_i)|\cdot|out(t_j)|)^{\frac{1}{2}}}$$

where

$$\delta(u,v) = \begin{cases} 1 & \text{if } (u,v) \text{ is aligned} \\ 0 & \text{otherwise} \end{cases}$$

4.5 Tree Structural Features

In addition to the lexical correspondence, we also capture the structural divergence by introducing the tree structural features as follows:

Span difference: Translational equivalent sub-tree pairs tend to share similar length of spans. Thus the model will penalize the candidate sub-tree pairs with largely different length of spans.

$$\xi(s_i, t_j) = \left|\frac{|in(s_i)|}{\max_{1\le k\le I}(|in(s_k)|)} - \frac{|in(t_j)|}{\max_{1\le h\le J}(|in(t_h)|)}\right|$$

Number of Descendants: Similarly, the number of the root's descendants of the aligned sub-trees should also correspond.

$$\tau(s_i, t_j) = \left|\frac{|R(s_i)|}{\max_{1\le k\le I}(|R(s_k)|)} - \frac{|R(t_j)|}{\max_{1\le h\le J}(|R(t_h)|)}\right|$$

where $R(.)$ refers to the descendant set of the root to an individual sub-tree.

Tree Depth difference: Intuitively, translational equivalent sub-tree pairs tend to have similar depth from the root node of the parse tree. We can further allow the model to penalize the candidate sub-tree pairs with different distance from the root node.

$$\omega(s_i, t_j) = \left| \frac{Depth(s_i)}{Height(S^I)} - \frac{Depth(t_j)}{Height(T^J)} \right|$$

4.6 Binary Grammatical Features

In the previous sections, we design some score based feature functions to describe syntactic tree structural similarities, rather than directly using the substructures. This is because for limited annotated tree alignment data, features like tokens and grammar rules are rather sparse. In spite of this, we still have a closed set of grammatical tags which can be covered by a small amount of data. Therefore, we use the combination of root grammar tags of the sub-tree pairs as binary features.

5 Training

We train the sub-tree alignment model in two steps:

Firstly, we learn the various feature functions. On one hand, GIZA++ is offline trained on a large amount of bilingual sentences to compute the lexical and word alignment features. On the other hand, the tree structural features, similar to word and phrase penalty features in phrase based SMT models, are computed online for both training and testing.

Secondly, we train the MaxEnt model in Eq. 1, using the training corpus which consists of the bilingual parse tree pairs with manually annotated sub-tree alignment. We apply the widely used GIS (Generalized Iterative Scaling) algorithm (Darroch and Ratcliff, 1972) to optimize λ_1^M. In practice, we modify Och's implementation YASMET.

Since we consider each sub-tree pair as an individual instance, it is easy to see that the negative samples heavily overwhelm the positive ones. For GIS training, such a skewed distribution easily drives the parameters to facilitate the negative instances. We address this problem by giving more weight to the positive training instances.

6 Experiments on Sub-Tree Alignments

We utilize two different corpora to evaluate the proposed sub-tree alignment method and its capability to plug in the related applications respectively. One is HIT English Chinese parallel tree bank with both tree structure and sub-tree alignment manually annotated. The other is the automatically parsed bilingual tree pairs (allowing minor parsing errors) with manually annotated sub-tree alignment. The latter benefits MT task, since most linguistically motivated syntax SMT systems require a held-out automatic parser to achieve rule induction.

6.1 Data preparation

For the gold standard corpus based experiment, we use HIT [1] Chinese-English parallel tree bank, which is collected from English learning text books in China as well as example sentences in dictionaries. It consists of 16131 gold standard parse tree pairs with manually annotated sub-tree alignments. The annotation strictly preserves the semantic equivalence, i.e., it only conducts *sure* links in the internal node level, while ignoring *possible* links adopted in word alignment. In contrast, in the POS level, n-to-n links are allowed in annotation. In order to be consistent with the definition in Section 2, we delete those n-to-n links in POS level. The word segmentation, tokenization and parse-tree in the corpus are manually constructed or checked. The Chinese parse tree in HIT tree bank adopts a different annotation criterion from the Penn TreeBank annotation, which is designed by the HIT research team. The new criterion can better facilitate the description of some rare structural phenomena in Chinese. The English parse tree still uses Penn TreeBank annotation. The statistics of HIT corpus is shown in Table 1.

	Chinese	English
# of Sentence pair	16131	
Avg. Sentence Length	13.06	13.00
Avg. # of sub-tree	21.60	23.74
Avg. # of alignment	11.71	

Table 1. Statistics for HIT gold standard Tree bank

Since the induction of sub-tree alignment is designed to benefit the machine translation modeling, it is preferable to conduct the sub-tree alignment experiment on the corpus for MT evaluation. However, most syntax based SMT systems use an automatic parser to facilitate training and decoding, which introduces parsing errors. Additionally, the gold standard HIT corpus is not applicable for MT

[1] HIT corpus is designed and constructed by HIT mitlab. http://mitlab.hit.edu.cn/index.php/resources.html . We licensed the corpus from them for research usage.

experiment due to problems of domain divergence, annotation discrepancy (Chinese parse tree adopts a different grammar from Penn Treebank annotations) and degree of tolerance for parsing errors.

Due to the above issues, we annotate a new data set to apply the sub-tree alignment in machine translation. We randomly select 300 bilingual sentence pairs from the Chinese-English FBIS corpus with the length ≤ 30 in both the source and target sides. The selected plain sentence pairs are further parsed by Stanford parser (Klein and Manning, 2003) on both the English and Chinese sides. We manually annotate the sub-tree alignment for the automatically parsed tree pairs according to the definition in Section 2. To be fully consistent with the definition, we strictly preserve the semantic equivalence for the aligned sub-trees to keep a high precision. In other words, we do not conduct any doubtful links. The corpus is further divided into 200 aligned tree pairs for training and 100 for testing. Some initial statistic of the automatically parsed corpus is shown in Table 2.

		Chinese	English
Train	# of Sentence pair	200	
	Avg. Sentence Length	17	20.84
	Avg. # of sub-tree	28.87	34.54
	Avg. # of alignment	17.07	
Test	# of Sentence pair	100	
	Avg. Sentence Length	16.84	20.75
	Avg. # of sub-tree	29.18	34.1
	Avg. # of alignment	17.75	

Table 2. FBIS selected Corpus Statistics

6.2 Baseline approach

We implement the work in Tinsley et al. (2007) as our baseline methodology.

Given a tree pair $< S^I, T^J >$, the baseline approach first takes all the links between the sub-tree pairs as alignment hypotheses, i.e., the Cartesian product of the two sub-tree sets:

$$\{s_1, \dots, s_i, \dots, s_I\} \times \{t_1, \dots, t_j, \dots, t_J\}$$

By using the lexical translation probabilities, each hypothesis is assigned an alignment score. All hypotheses with zero score are pruned out. Then the algorithm iteratively selects the link of the sub-tree pairs with the maximum score as a *sure* link, and blocks all hypotheses that contradict with this link and itself, until no non-blocked hypotheses remain.

The baseline system uses many heuristics in searching the optimal solutions with alternative score functions. Heuristic *skip1* skips the tied hypotheses with the same score, until it finds the highest-scoring hypothesis with no competitors of the same score. Heuristic *skip2* deals with the same problem. Initially, it skips over the tied hypotheses. When a hypothesis sub-tree pair (s_i, t_j) without any competitor of the same score is found, where neither s_i nor t_j has been skipped over, the hypothesis is chosen as a *sure* link. Heuristic *span1* postpones the selection of the hypotheses on the POS level. Since the highest-scoring hypotheses tend to appear on the leaf nodes, it may introduce ambiguity when conducting the alignment for a POS node whose child word appears twice in a sentence.

The baseline method proposes two score functions based on the lexical translation probability. They also compute the score function by splitting the tree into the internal and external components.

Tinsley et al. (2007) adopt the lexical translation probabilities dumped by GIZA++ (Och and Ney, 2003) to compute the span based scores for each pair of sub-trees. Although all of their heuristics combinations are re-implemented in our study, we only present the best result among them with the highest Recall and F-value as our baseline, denoted as *skip2_s1_span1*[2].

6.3 Experimental settings

- To examine the effectiveness of the proposed features, we

(1) learn the word alignment using the combination of the 14k of HIT tree bank and FBIS (240k) corpus for both our approach and the baseline method, and divide the remaining HIT corpus as 1k for training and 1k for testing.

(2) learn the word alignment on the entire FBIS training corpus (240k) for both our approach and the baseline method. We then train and test on FBIS corpus of 200 and 100 respectively as stated in Table 2.

- In our task, annotating large amount of sub-tree alignment corpus is time consuming and more difficult compared with the tasks like sequence labeling. One of the important issues we are concerned about is whether we can achieve an acceptable performance with limited training data. We

(3) adopt the entire FBIS data (240k) to learn the word alignment and various amount of HIT gold standard corpus to train the MaxEnt model. Then we test the alignment performance on the same HIT test set (1k) as (1).

[2] s1 denotes score function 1 in Tinsley et al. (2007)

Features	Precision	Recall	F-value
In Lexical	50.96	48.11	49.49
+ InOut Lexical	55.26	53.84	54.54
+ In word align	56.16	60.59	58.29
+ InOut word align	55.80	62.25	58.85
+ Tree Structure	57.64	63.11	60.25
+ Binary Feature	73.14	85.11	78.67
Baseline [Tinsley 2007]	64.14	66.99	65.53

Table 3. Sub-tree alignment of different feature combination for HIT gold standard test set

Features	Precision	Recall	F-value
In Lexical	63.53	54.87	58.88
+ InOut Lexical	66.00	63.66	64.81
+ In word align	70.89	75.88	73.30
+ InOut word align	72.05	80.16	75.89
+ Tree Structure	72.03	80.95	76.23
+ Binary Feature	76.08	85.29	80.42
Baseline [Tinsley 2007]	70.48	78.70	74.36

Table 4. Sub-tree alignment of different feature combination for FBIS test set

• We further test the robustness of our method under different amount of data to learn the lexical and word alignment feature functions. We gradually change the amount of FBIS corpus to train the word alignment. Then we

(4) use the same training (1k) and testing data (1k) with (1);

(5) use FBIS corpus 200 to train MaxEnt model and 100 for testing similar to (2).

6.4 Experimental results

We use Precision, Recall and F-score to measure the alignment performance and obtain the results as follows:

• In Table 3 and 4 for **Exp (1)** and **(2)** respectively, we show that by incrementally adding new features in a certain order, the F-value consistently increases and both outperform the baseline method. From both tables, we find that the **Binary features**, with the combination of root grammar tags of the sub-tree pairs, significantly improve the alignment performance. We also try the different combinations of the parent, child or even siblings to the root nodes. However, all these derivative configurations decrease the performance. We attribute the ineffectiveness to data sparseness. Further exploration suggests that the binary feature in HIT gold standard corpus exhibits a substantially larger improvement against other features than FBIS corpus (Table 3 against Table 4). The reason could be that the grammar tags in the gold standard corpus are accurate, while FBIS corpus suffers from parsing errors. Apart from that, the lexical/word-alignment features in Table 3 do not perform well, since the word alignment is trained mainly on the cross domain FBIS corpus. This is also an important reason why there is a large gap in performance between Table 3 and 4, where the automatic parsed FBIS corpus performs better than HIT gold standard tree bank in all configurations as well as the baseline.

• In Fig. 2(a) for **Exp (3)**, we examine performance under different amount of training data from 1k to 15k. The results change very little with over the amount of 1k. Even with only 0.25k training data, we are able to gain a result close to the best performance. This suggests that by utilizing only a small amount of sub-tree aligned corpus, we can still achieve a satisfactory alignment result. The benefits come from the usage of the score based feature functions by avoiding using sub-structures as binary features, which suffers from the data sparseness problem.

• In Fig. 2(b-e) for **Exp (4&5)**, we find that increasing the amount of corpus to train GIZA++ does not improve much for the proposed method on both HIT gold standard corpus (Fig. 2: b, c) and the automatic parsed data (Fig. 2: d, e). This is due to the various kinds of features utilized by the MaxEnt model, which does not bet on the lexical and word alignment feature too much. As for the baseline method, we can only detect a relatively large improvement in the initial increment of corpus, while later additions do not help. This result suggests that the baseline method is relatively less extensible since it works completely on the lexical similarities which can be only learned from the word alignment corpus.

7 Experiments on Machine Translation

In addition to the alignment evaluation, we conduct MT evaluation as well. We explore the effectiveness of sub-tree alignment for both phrase and linguistically motivated syntax based systems.

7.1 Experimental configuration

In the experiments, we train the translation model on FBIS corpus (7.2M (Chinese) + 9.2M (English) words in 240,000 sentence pairs) and train a 4-gram language model on the Xinhua portion of the English Gigaword corpus (181M words) using the SRILM Toolkits (Stolcke, 2002). We use these

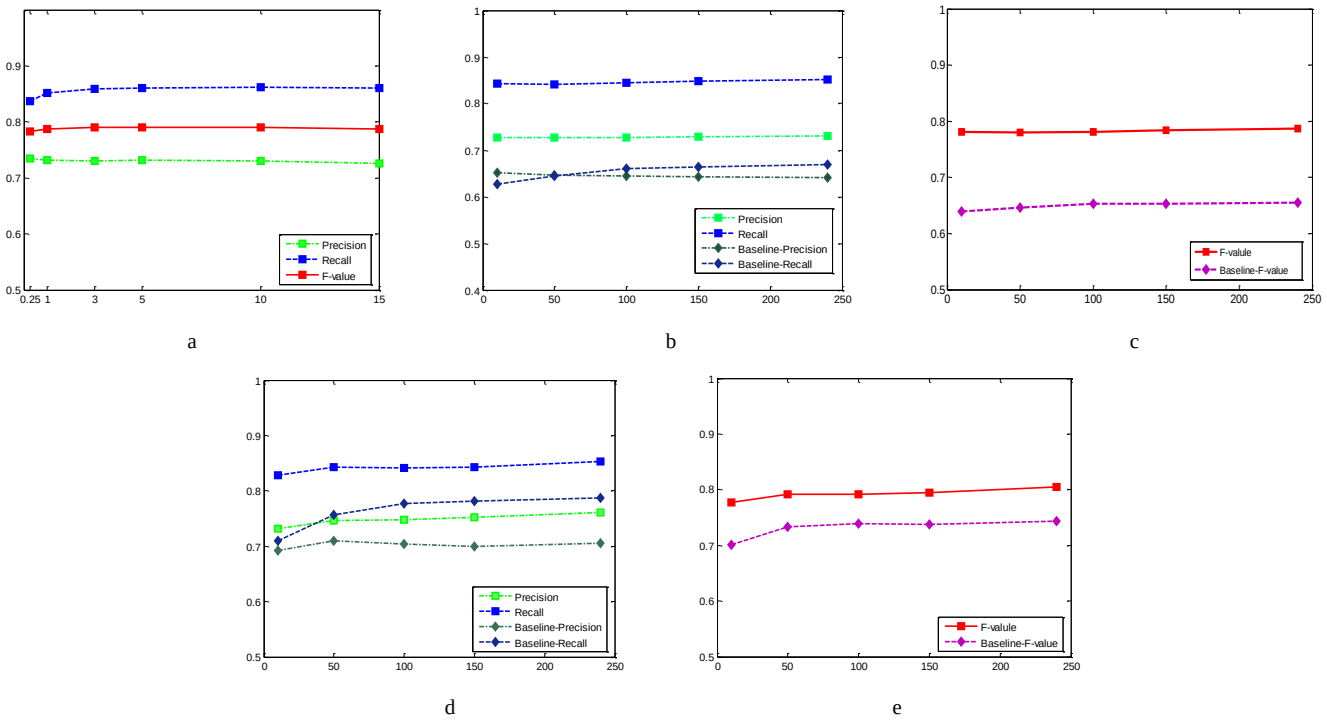


Figure 2: a. Precision/Recall/F-score for various amount of training data (k).
b~e. Various amount of data to train word alignment
b. Precision/Recall for HIT test set. c. F-score for HIT test set.
d. Precision/Recall for FBIS test set. e. F-score for FBIS test set.

sentences with less than 50 characters from the NIST MT-2002 test set as the development set (to speed up tuning for syntax based system) and the NIST MT-2005 test set as our test set. We use the Stanford parser (Klein and Manning, 2003) to parse bilingual sentences on the training set and Chinese sentences on the development and test set. The evaluation metric is case-sensitive BLEU-4.

For the phrase based system, we use Moses (Koehn et al, 2007) with its default settings. For the syntax based system, since sub-tree alignment can directly benefit Tree-2-Tree based systems, we apply the sub-tree alignment in an SMT system based on Synchronous Tree Substitution Grammar (STSG) (Zhang et al., 2007). The STSG based decoder uses a pair of *elementary tree* as a basic translation unit. Recent research on tree based systems shows that relaxing the restriction from tree structure to tree sequence structure (Synchronous Tree Sequence Substitution Grammar: STSSG) significantly improves the translation performance (Zhang et al., 2008). We implement the STSG/STSSG based model in Pisces decoder with the same features and settings in Sun et al. (2009). The STSSG based decoder translates each span iteratively in a bottom up manner which guarantees that when translating a source span, any of its sub-spans has already been translated. The STSG based experiment can be easily achieved by restricting the translation rule set in the STSSG decoder to be elementary tree pairs only.

For the alignment setting of the baselines, we use the word alignment trained on the entire FBIS(240k) corpus by GIZA++ with heuristic grow-diag-final for Moses and the syntax systems and perform rule extraction constrained on the word alignment. As for the experiments adopting sub-tree alignment, we use the above word alignment to learn lexical/word alignment features, and train the sub-tree alignment model with FBIS training data (200).

7.2 Experimental results

Utilizing the syntactic rules only has been argued to be ineffective (Koehn et al., 2003). Therefore, instead of using the sub-tree aligned rules only, we try to improve the word alignment constrained rule set by sub-tree alignment as shown in Table 5.

Firstly, we try to *Directly Concatenate* (DirC) the sub-tree alignment constraint rule set[3] to the original syntax/phrase rule set based on word alignment. Then we re-train the MT model based

[3] For syntax based system, it's just the sub-tree pairs deducted from the sub-tree alignment; for phrase based system, it's the phrases with context equivalent to the aligned sub-tree pairs.

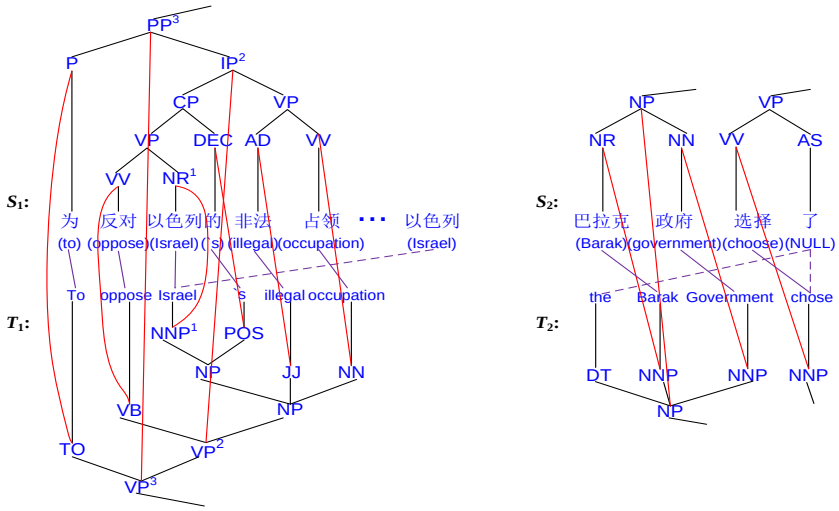


Figure 3: Comparison between Sub-tree alignment results and Word alignment results

on the obtained rule set. Tinsley et al. (2009) attempts different duplication of sub-tree alignment constraint rule set to append to the original phrase rule set and reports positive results. However, as shown in Table 5, we only achieve very minor improvement (in STSSG based model the score even drops) by direct introducing the new rules.

Secondly, we propose a new approach to utilize sub-tree alignment by modifying the rule extraction process. We allow the bilingual phrases which are consistent with *Either* Word alignment *or* Sub-tree alignment (EWoS) instead of to be consistent with word alignment only. The results in Table 5 show that EWoS achieves consistently better performance than the baseline and DirC method. We also find that sub-tree alignment benefits the STSSG based model less compared with other systems. This is probably due to the fact that the STSSG based system relies much on the tree sequence rules.

To benefit intuitive understanding, we provide two alignment snippets in the MT training corpus in Fig. 3, where the red lines across the non-terminal nodes are the sub-tree aligned links conducted by our model, while the purple lines across the terminal nodes are the word alignment links trained by GIZA++. In the first example, the word *Israel* is wrongly aligned to two "以色列"s by GIZA++, where the wrong link is denoted by the dash line. This is common, since in a compound sentence in English, the entities appeared more than once are often replaced by pronouns at its later appearances. Therefore, the syntactic rules constraint by NR^1-NNP^1, IP^2-VP^2 and PP^3-VP^3 respectively cannot be extracted for syntax systems; while for phrase systems, context around the first "以色列" cannot be fully explored. In the second example, the empty word "了" is wrongly aligned, which usually occurs in Chinese-English word alignment. As shown in Fig. 3, both cases can be resolved by sub-tree alignment conducted by our model, indicating that sub-tree alignment is a decent supplement to the word alignment rule set.

System	Rules	BLEU
Moses	BP*	23.86
	DirC	24.12
	EWoS	24.45
Syntax STSG	STSG	24.71
	DirC	24.91
	EWoS	25.21
Syntax STSSG	STSSG	25.92
	DirC	25.88
	EWoS	26.12

Table 5. MT evaluation on various systems
BP* denotes bilingual phrases.
BP, STSG, STSSG are baseline rule sets using word alignment to constrain rule extraction.

8 Conclusion

In this paper, we propose a framework for bilingual sub-tree alignment using Maximum Entropy model. We explore various lexical and structural features to improve the alignment performance. We also manually annotated the automatic parsed tree pairs for both alignment evaluation and MT experiment. Experimental results show that our alignment framework significantly outperforms the baseline method and the proposed features are very effective to capture the bilingual structural similarity. Additionally, we find that our approach can perform well using only a small amount of sub-tree aligned training corpus. Further experiment shows that our approach benefits both phrase and syntax based MT systems.

References

David Burkett and Dan Klein. 2008. *Two languages are better than one* (*for syntactic parsing*). In Proceedings of EMNLP-08. 877-886.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang and Ignacio Thayer. 2006. *Scalable Inference and training of context-rich syntactic translation models*. In Proceedings of COLING-ACL-06. 961-968.

Jonathan Graehl and Kevin Knight. 2004. *Training tree transducers*. In Proceedings of HLT-NAACL-2004. 105-112.

Declan Groves, Mary Hearne, and Andy Way. 2004. *Robust sub-sentential alignment of phrase-structure trees*. In Proceedings of COLING-04, pages 1072-1078.

Liang Huang, Kevin Knight and Aravind Joshi. 2006. *Statistical syntax-directed translation with extended domain of Locality*. In Proceedings of AMTA-06.

Kenji Imamura. 2001. *Hierarchical Phrase Alignment Harmonized with Parsing*. In Proceedings of NLPRS. 377-384.

Dan Klein and Christopher D. Manning. 2003. *Accurate Unlexicalized Parsing*. In Proceedings of ACL-03. 423-430.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin and Evan Herbst. 2007. *Moses: Open Source Toolkit for Statistical Machine Translation*. In Proceedings of ACL-07. 177-180.

Philipp Koehn, Franz Josef Och and Daniel Marcu. 2003. *Statistical Phrase-based Translation*. In Proceedings of HLT-NAACL-2003. 48-54.

Yang Liu, Qun Liu and Shouxun Lin. 2006. *Tree-to-String alignment template for statistical machine translation*. In Proceedings of ACL-06, 609-616.

Daniel Marcu, Wei Wang, Abdessamad Echihabi and Kevin Knight. 2006. *SPMT: statistical machine translation with syntactified target language phrases*. In Proceedings of EMNLP-06. 44-52.

Franz Josef Och and Hermann Ney. 2003. *A systematic comparison of various statistical alignment models*. Computational Linguistics, 29(1):19-51, March.

Andreas Stolcke. 2002. *SRILM - an extensible language modeling toolkit*. In Proceedings of ICSLP-02. 901-904.

Jun Sun, Min Zhang and Chew Lim Tan. 2009. *A non-contiguous Tree Sequence Alignment-based Model for Statistical Machine Translation*. In Proceedings of ACL-IJCNLP-09. 914-922.

John Tinsley, Ventsislav Zhechev, Mary Hearne, and Andy Way. 2007. *Robust language pair-independent sub-tree alignment*. In Proceedings of Machine Translation Summit-XI-07.

John Tinsley, Mary Hearne, and Andy Way. 2009. *Parallel treebanks in phrase-based statistical machine translation*. In Proceedings of CICLING-09.

Min Zhang, Hongfei Jiang, AiTi Aw, Jun Sun, Sheng Li and Chew Lim Tan. 2007. *A tree-to-tree alignment-based model for statistical machine translation*. In Proceedings of MT Summit-XI -07. 535-542.

Min Zhang, Hongfei Jiang, AiTi Aw, Haizhou Li, Chew Lim Tan and Sheng Li. 2008. *A tree sequence alignment-based tree-to-tree translation model*. In Proceedings of ACL-08. 559-567.

Investigating the cross-linguistic potential of VerbNet -style classification

Lin Sun and Anna Korhonen
Computer Laboratory
University of Cambridge
ls418,alk23@cl.cam.ac.uk

Thierry Poibeau
LaTTiCe, UMR8094
CNRS & ENS
thierry.poibeau@ens.fr

Cédric Messiant
LIPN, UMR7030
CNRS & U. Paris 13
cedric.messiant@lipn.fr

Abstract

Verb classes which integrate a wide range of linguistic properties (Levin, 1993) have proved useful for natural language processing (NLP) applications. However, the real-world use of these classes has been limited because for most languages, no resources similar to VerbNet (Kipper-Schuler, 2005) are available. We apply a verb clustering approach developed for English to French – a language for which no such experiment has been conducted yet. Our investigation shows that not only the general methodology but also the best performing features are transferable between the languages, making it possible to learn useful VerbNet style classes for French automatically without language-specific tuning.

1 Introduction

A number of verb classifications have been built to support natural language processing (NLP) tasks (Grishman et al., 1994; Miller, 1995; Baker et al., 1998; Palmer et al., 2005; Kipper-Schuler, 2005; Hovy et al., 2006). These include both syntactic and semantic classifications, as well as ones which integrate aspects of both. Classifications which integrate a wide range of linguistic properties can be particularly useful for NLP applications suffering from data sparseness. One such classification is VerbNet (Kipper-Schuler, 2005). Building on the taxonomy of Levin (1993), VerbNet groups verbs (e.g. *deliver, post, dispatch*) into classes (e.g. SEND) on the basis of their shared meaning components and syntactic behaviour, identified in terms of meaning preserving diathesis alternations. Such classes can be identified across the entire lexicon, and they may also apply across languages, since their meaning components are said to be cross-linguistically applicable (Jackendoff, 1990).

Offering a powerful tool for generalization, abstraction and prediction, VerbNet classes have been used to support many important NLP tasks, including e.g. computational lexicography, parsing, word sense disambiguation, semantic role labeling, information extraction, question-answering, and machine translation (Swier and Stevenson, 2004; Dang, 2004; Shi and Mihalcea, 2005; Abend et al., 2008). However, to date their exploitation has been limited because for most languages, no Levin style classification is available.

Since manual classification is costly (Kipper et al., 2008) automatic approaches have been proposed recently which could be used to learn novel classifications in a cost-effective manner (Joanis et al., 2008; Li and Brew, 2008; Ó Séaghdha and Copestake, 2008; Vlachos et al., 2009; Sun and Korhonen, 2009). However, most work on Levin type classification has focussed on English. Large-scale research on other languages such as German (Schulte im Walde, 2006) and Japanese (Suzuki and Fukumoto, 2009) has focussed on semantic classification. Although the two classification systems have shared properties, studies comparing the overlap between VerbNet and WordNet (Miller, 1995) have reported that the mapping is only partial and many to many due to fine-grained nature of classes based on synonymy (Shi and Mihalcea, 2005; Abend et al., 2008).

Only few studies have been conducted on Levin style classification for languages other than English. In their experiment involving 59 verbs and three classes, Merlo et al. (2002) applied a supervised approach developed for English to Italian, obtaining high accuracy (86.3%). In another experiment with 60 verbs and three classes,

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1056–1064,
Beijing, August 2010

they showed that features extracted from Chinese translations of English verbs can improve English classification. These results are promising, but those from a later experiment by Ferrer (2004) are not. Ferrer applied a clustering approach developed for English to Spanish, and evaluated it against the manual classification of Vázquez et al. (2000), constructed using criteria similar (but not identical) to Levin's. This experiment involving 514 verbs and 31 classes produced results only slightly better than the random baseline.

In this paper, we investigate the cross-linguistic potential of Levin style classification further. In past years, verb classification techniques – in particular unsupervised ones – have improved considerably, making investigations for a new language more feasible. We take a recent verb clustering approach developed for English (Sun and Korhonen, 2009) and apply it to French – a major language for which no such experiment has been conducted yet. Basic NLP resources (corpora, taggers, parsers and subcategorization acquisition systems) are now sufficiently developed for this language for the application of a state-of-the-art verb clustering approach to be realistic.

Our investigation reveals similarities between the English and French classifications, supporting the linguistic hypothesis (Jackendoff, 1990) and the earlier result of Merlo et al. (2002) that Levin classes have a strong cross-linguistic basis. Not only the general methodology but also best performing features are transferable between the languages, making it possible to learn useful classes for French automatically without language-specific tuning.

2 French Gold Standard

The development of an automatic verb classification approach requires at least an initial gold standard. Some syntactic (Gross, 1975) and semantic (Vossen, 1998) verb classifications exist for French, along with ones which integrate aspects of both (Saint-Dizier, 1998). Since none of these resources offer classes similar to Levins', we followed the idea of Merlo et al. (2002) and translated a number of Levin classes from English to French. As our aim was to to investigate the cross-linguistic applicability of classes, we took an English gold standard which has been used to evaluate several recent clustering works – that of Sun et al. (2008). This resource includes 17 fine-grained Levin classes. Each class has 12 member verbs whose predominant sense in English (according to WordNet) belongs to that class.

Member verbs were first translated to French. Where several relevant translations were identified, each of them was considered. For each candidate verb, subcategorization frames (SCFs) were identified and diathesis alternations were considered using the criteria of Levin (1993): alternations must result in the same or extended verb sense. Only verbs sharing diathesis alternations were kept in the class.

For example, the gold standard class 31.1 AMUSE includes the following English verbs: *stimulate, threaten, shock, confuse, upset, overwhelm, scare, disappoint, delight, exhaust, intimidate* and *frighten*. Relevant French translations were identified for all of them: *abattre, accabler, briser, déprimer, consterner, anéantir, épuiser, exténuer, écraser, ennuyer, éreinter, inonder*. The majority of these verbs take similar SCFs and diathesis alternations, e.g. *Cette affaire écrase Marie (de chagrin), Marie est écrasée par le chagrin, Le chagrin écrase Marie*. However, *stimuler* (*stimulate*) and *menacer* (*threaten*) do not, and they were therefore removed.

40% of translations were discarded from classes because they did not share the same alernations. The final version of the gold standard (shown in table 1) includes 171 verbs in 16 classes. Each class is named according to the original Levin class. The smallest class (30.3) includes 7 verbs and the largest (37.3) 16. The average number of verbs per class is 10.7.

3 Verb Clustering

We performed an experiment where we

- took a French corpus and a SCF lexicon automatically extracted from that corpus,
- extracted from these resources a range of features (lexical, syntactic and semantic) – a representative sample of those employed in recent English experiments,

Class No	Class	Verbs
9.1	PUT	accrocher, déposer, mettre, placer, répartir, réintégrer, empiler, emporter, enfermer, insérer, installer
10.1	REMOVE	ôter, enlever, retirer, supprimer, retrancher, débarrasser, soustraire, décompter, éliminer
11.1	SEND	envoyer, lancer, transmettre, adresser, porter, expédier, transporter, jeter, renvoyer, livrer
13.5.1	GET	acheter, prendre, saisir, réserver, conserver, garder, préserver, maintenir, retenir, louer, affréter
18.1	HIT	cogner, heurter, battre, frapper, fouetter, taper, rosser, brutaliser, éreinter, maltraiter, corriger,
22.2	AMALGAMATE	incorporer, associer, réunir, mélanger, mêler, unir, assembler, combiner, lier, fusionner
29.2	CHARACTERIZE	appréhender, concevoir, considérer, décrire, définir, dépeindre, désigner, envisager, identifier, montrer, percevoir, représenter, ressentir
30.3	PEER	regarder, écouter, examiner, considérer, voir, scruter, dévisager
31.1	AMUSE	abattre, accabler, briser, déprimer, consterner, anéantir, épuiser, exténuer, écraser, ennuyer, éreinter, inonder,
36.1	CORRESPOND	coopérer, participer, collaborer, concourir, contribuer, prendre part, s'associer, travaille
37.3	MANNER OF SPEAKING	râler, gronder, crier, ronchonner, grogner, bougonner, maugréer, rouspéter, grommeler, larmoyer, gémir, geindre, hurler, gueuler, brailler, chuchoter
37.7	SAY	dire, révéler, déclarer, signaler, indiquer, montrer, annoncer, répondre, affirmer, certifier, répliquer
43.1	LIGHT EMISSION	briller, étinceler, flamboyer, luire, resplendir, pétiller, rutiler, rayonner., scintiller
45.4	CHANGE OF STATE	mélanger, fusionner, consolider, renforcer, fortifier, adoucir, polir, atténuer, tempérer, pétrir, façonner, former
47.3	MODES OF BEING	trembler, frémir, osciller, vaciller, vibrer, tressaillir, frissonner, palpiter, grésiller, trembloter, palpiter
51.3.2	RUN	voyager, aller, se promener, errer, circuler, se déplacer, courir, bouger, naviguer, passer

Table 1: A Levin style gold standard for French

- clustered the features using a method which has proved promising in both English and German experiments: spectral clustering,
- evaluated the clusters both quantitatively (using the gold standard) and qualitatively,
- and compared the performance to that recently obtained for English in order to gain a better understanding of the cross-linguistic and language-specific properties of verb classification

This work is described in the subsequent sections.

3.1 Data: the LexSchem Lexicon

We extracted the features for clustering from LexSchem (Messiant et al., 2008). This large subcategorization lexicon provides SCF frequency information for 3,297 French verbs. It was acquired fully automatically from Le Monde newspaper corpus (200M words from years 1991-2000) using ASSCI – a recent subcategorization acquisition system for French (Messiant, 2008). Systems similar to ASSCI have been used in recent verb classification works e.g. (Schulte im Walde, 2006; Li and Brew, 2008; Sun and Korhonen, 2009). Like these other systems, ASSCI takes raw corpus data as input. The data is first tagged and lemmatized using the Tree-Tagger and then parsed using Syntex (Bourigault et al., 2005). Syntex is a shallow parser which employs a combination of statistics and heuristics to identify grammatical relations (GRs) in sentences. ASSCI considers GRs where the target verbs occur and constructs SCFs from nominal, prepositional and adjectival phrases, and infinitival and subordinate clauses. When a verb has no dependency, its SCF is considered as intransitive. ASSCI assumes no predefined list of SCFs but almost any combination of permitted constructions can appear as a candidate SCF. The number of automatically generated SCF types in LexSchem is 336.

Many candidate SCFs are noisy due to processing errors and the difficulty of argument-adjunct distinction. Most SCF systems assume that true arguments occur in argument positions more frequently than adjuncts. Many systems also integrate filters for removing noise from system output. When LexSchem was evaluated after filter-

ing its F-measure was 69 – which is similar to that of other current SCF systems (Messiant et al., 2008) We used the unfiltered version of the lexicon because English experiments have shown that information about adjuncts can help verb clustering (Sun et al., 2008).

4 Features

Lexical entries in LexSchem provide a variety of material for verb clustering. Using this material, we constructed a range of features for experimentation. The first three include basic information about SCFs:

F1: SCFs and their relative frequencies with individual verbs. SCFs abstract over particles and prepositions.

F2: F1, with SCFs parameterized for the tense (the POS tag) of the verb.

F3: F2, with SCFs parameterized for prepositions (PP).

The following six features include information about the lexical context (co-occurrences) of verbs. We adopt the best method of Li and Brew (2008) where collocations (COs) are extracted from the window of words immediately preceding and following a lemmatized verb. Stop words are removed prior to extraction.

F4, F6, F8: COs are extracted from the window of 4, 6 and 8 words, respectively. The relative word position is ignored.

F5, F7, F9: F4, F6 and F8 with the relative word position recorded.

The next four features include information about lexical preferences (LP) of verbs in argument head positions of specific GRs associated with the verb:

F10: LP(PREP): the type and frequency of prepositions in the preposition (PREP) relation.

F11: LP(SUBJ): the type and frequency of nouns in the subject (SUBJ) relation.

F12: LP(IOBJ): the type and frequency of nouns in the object (OBJ) and indirect object (IOBJ) relation.

F13: LP(ALL): the combination of F10-F13.

The final two features refine SCF features with LPs and semantic information about verb selectional preferences (SP):

F14-F16: F1-F3 parameterized for LPs.

F17: F3 refined with SPs.

We adopt a fully unsupervised approach to SP acquisition using the method of Sun and Korhonen (2009), with the difference that we determine the optimal number of SP clusters automatically following Zelnik-Manor and Perona (2004). The method is introduced in the following section. The approach involves (i) taking the GRs (SUBJ, OBJ, IOBJ) associated with verbs, (ii) extracting all the argument heads in these GRs, and (iii) clustering the resulting N most frequent argument heads into M classes. The empirically determined N 200 was used. The method produced 40 SP clusters.

5 Clustering Methods

Spectral clustering (SPEC) has proved promising in previous verb clustering experiments (Brew and Schulte im Walde, 2002; Sun and Korhonen, 2009) and other similar NLP tasks involving high dimensional feature space (Chen et al., 2006). Following Sun and Korhonen (2009) we used the MNCut spectral clustering (Meila and Shi, 2001) which has a wide applicability and a clear probabilistic interpretation (von Luxburg, 2007; Verma and Meila, 2005). However, we extended the method to determine the optimal number of clusters automatically using the technique proposed by (Zelnik-Manor and Perona, 2004).

Clustering groups a given set of verbs $V = \{v_n\}_{n=1}^{N}$ into a disjoint partition of K classes. SPEC takes a similarity matrix as input. All our features can be viewed as probabilistic distributions because the combination of different features is performed via parameterization. Thus we use the Jensen-Shannon divergence (JSD) to construct the similarity matrix. The JSD between

two feature vectors v and v' is $d_{jsd}(v, v') = \frac{1}{2}D(v||m) + \frac{1}{2}D(v'||m)$ where D is the Kullback-Leibler divergence, and m is the average of the v and v'.

The similarity matrix W is constructed where $W_{ij} = \exp(-d_{jsd}(v, v'))$. In SPEC, the similarities W_{ij} are viewed as the connection weight ij of a graph G over V. The similarity matrix W is thus the adjacency matrix for G. The degree of a vertex i is $d_i = \sum_{j=1}^{N} w_{ij}$. A cut between two partitions A and A' is defined to be $\text{Cut}(A, A') = \sum_{m \in A, n \in A'} W_{mn}$.

The similarity matrix W is normalized into a stochastic matrix P.

$$P = D^{-1}W \quad (1)$$

The degree matrix D is a diagonal matrix where $D_{ii} = d_i$.

It was shown by Meila and Shi (2001) that if P has the K leading eigenvectors that are piecewise constant[1] with respect to a partition I^* and their eigenvalues are not zero, then I^* minimizes the multiway normalized cut(MNCut):

$$\text{MNCut}(I) = K - \sum_{k=1}^{K} \frac{\text{Cut}(I_k, I_k)}{\text{Cut}(I_k, I)}$$

P_{mn} can be interpreted as the transition probability between vertices m, n. The criterion can thus be expressed as $\text{MNCut}(I) = \sum_{k=1}^{K}(1 - P(I_k \rightarrow I_k|I_k))$ (Meila, 2001), which is the sum of transition probabilities across different clusters. This criterion finds the partition where the random walks are most likely to happen within the same cluster. In practice, the leading eigenvectors of P are not piecewise constant. But we can extract the partition by finding the approximately equal elements in the eigenvectors using a clustering algorithm like K-Means.

As the value of K is not known beforehand, we use Zelnik-Manor and Perona (2004)'s method to estimate it. This method finds the optimal value by minimizing a cost function based on the eigenvector structure of W.

Like Brew and Schulte im Walde (2002), we compare SPEC against a K-Means baseline. We used the Matlab implementation with euclidean distance as the distance measure.

[1]The eigenvector v is piecewise constant with respect to I if $v(i) = v(j) \forall i, j \in I_k$ and $k \in 1, 2...K$

6 Experimental Evaluation

6.1 Data and Pre-processing

The SCF-based features (F1-F3 and F14-F17) were extracted directly from LexSchem. The CO (F4-F9) and LP features (F10-F13) were extracted from the raw and parsed corpus sentences, respectively, which were used for creating the lexicon. Features that only appeared once were removed. Feature vectors were normalized by the sum of the feature values before clustering. Since our clustering algorithms have an element of randomness, we repeated clustering multiple times. We report the results that minimize the distortion (the distance to cluster centroid).

6.2 Evaluation Measures

We employ the same measures for evaluation as previously employed e.g. by Ó Séaghdha and Copestake (2008) and Sun and Korhonen (2009).

The first measure is modified purity (mPUR) – a global measure which evaluates the mean precision of clusters. Each cluster is associated with its prevalent class. The number of verbs in a cluster K that take this class is denoted by $n_{prevalent}$(K). Verbs that do not take it are considered as errors. Clusters where $n_{prevalent}$(K) = 1 are disregarded as not to introduce a bias towards singletons:

$$\text{mPUR} = \frac{\sum_{n_{prevalent(k_i)>2}} n_{prevalent(k_i)}}{\text{number of verbs}}$$

The second measure is weighted class accuracy (ACC): the proportion of members of dominant clusters DOM-CLUST$_i$ within all classes c_i.

$$\text{ACC} = \frac{\sum_{i=1}^{C} \text{verbs in DOM-CLUST}_i}{\text{number of verbs}}$$

mPUR and ACC can be seen as a measure of precision(P) and recall(R) respectively. We calculate F measure as the harmonic mean of P and R:

$$\text{F} = \frac{2 \cdot \text{mPUR} \cdot \text{ACC}}{\text{mPUR} + \text{ACC}}$$

The random baseline (BL) is calculated as follows: $\text{BL} = 1/\text{number of classes}$

7 Evaluation

7.1 Quantitative Evaluation

In our first experiment, we evaluated 116 verbs – those which appeared in LexSchem the minimum

of 150 times. We did this because English experiments had shown that due to the Zipfian nature of SCF distributions, 150 corpus occurrences are typically needed to obtain a sufficient number of frames for clustering (Sun et al., 2008).

Table 2 shows F-measure results for all the features. The 4th column of the table shows, for comparison, the results of Sun and Korhonen (2009) obtained for English when they used the same features as us, clustered them using SPEC, and evaluated them against the English version of our gold standard, also using F-measure[2].

As expected, SPEC (the 2nd column) outperforms K-Means (the 3rd column). Looking at the basic SCF features F1-F3, we can see that they perform significantly better than the BL method. F3 performs the best among the three features both in French (50.6 F) and in English (63.3 F). We therefore use F3 as the SCF feature in F14-F17 (the same was done for English).

In French, most CO features (F4-F9) outperform SCF features. The best result is obtained with F7: 55.1 F. This is clearly better than the best SCF result 50.6 (F3). This result is interesting since SCFs correspond better than COs with features used in manual Levin classification. Also, SCFs perform considerably better than COs in the English experiment (we only have the result for F4 available, but it is considerably lower than the result for F3). However, earlier English studies have reported contradictory results (e.g. Li and Brew (2008) showed that CO performs better than SCF in supervised verb classification), indicating that the role of CO features in verb classification requires further investigation.

Looking at the LP features, F13 produces the best F (52.7) for French which is slightly better than the best SCF result for the language. Also in English, F13 performs the best in this feature group and yields a higher result than the best SCF-based feature F3.

Parameterizing the best SCF feature F3 with LPs (F14-16) and SPs (F17) yields better performance in French. F15 and F17 have the F of 54.5 and 54.6, respectively. These results are so close to the result of the best CO feature F7 (55.1 – which is the highest result in this experiment) that the differences are not statistically significant. In English, the results of F14-F17 are similarly good; however, only F17 beats the already high performance of F13.

On the basis of this experiment, it is difficult to tell whether shallow CO features or more sophisticated SCF-based features are better for French. In the English experiment sophisticated features performed better (the SCF-SP feature was the best). However, the English experiment employed a much larger dataset. These more sophisticated features may suffer from data sparseness in our French experiment since although we required the minimum of 150 occurrences per verb, verb clustering performance tends to improve when more data is available, and given the fine-grained nature of LexShem SCFs it is likely that more data is required for optimal performance.

We therefore performed another experiment with French on the full set of 147 verbs, using SPEC, where we investigated the effect of instance filtering on the performance of the best features from each feature group: F3, F7, F13 and F17. The results shown in Table 3 reveal that the performance of the features remains fairly similar until the instance threshold of 1000. When 2000 occurrences per verb are used, the differences become clearer, until at the threshold of 4000, it is obvious that the most sophisticated SCF-SP feature F17 is by far the best feature for French (65.4 F) and the SCF feature F3 the second best (60.5 F). The CO-feature F7 and the LP feature F13 are not nearly as good (53.4 and 51.0 F).

Although the results at different thresholds are not comparable due to the different number of verbs and classes (see columns 2-3), the results for features at the same threshold are. Those results suggest that when 2000 or more occurrences per verb are used, most features perform like they performed for English in the experiment of Sun and Korhonen (2009), with CO being the least informative[3] and SCF-SP being the most informa-

[2]Note that the results for the two languages are not mutually comparable due to differences in test sets, data sizes, and feature extraction systems (see Section 8 for discussion). The results for English are included so that we can compare the relative performance of individual features in the two languages in question.

[3]However, it is worth noting that CO is not a useless feature. As table 3 shows, when 150 or fewer occurrences are

		SPEC	K	Eng.
BL		6.7	6.7	6.7
F1	SCF	42.4	39.3	57.8
F2	SCF(POS)	45.9	40.3	46.7
F3	SCF(PP)	**50.6**	36.9	63.3
F4	CO(4)	50.3	38.2	40.9
F5	CO(4+loc)	48.8	26.3	-
F6	CO(6)	52.7	29.2	-
F7	CO(6+loc)	**55.1**	33.8	-
F8	CO(8)	54.2	36.4	-
F9	CO(8+loc)	54.6	37.2	-
F10	LP(PREP)	35.5	32.8	49.0
F11	LP(SUBJ)	33.7	23.6	-
F12	LP(OBJ)	50.1	33.3	-
F13	LP(ALL)	**52.7**	40.1	74.6
F14	SCF+LP(SUBJ)	50.3	40.1	71.7
F15	SCF+LP(OBJ)	**54.5**	35.6	74.0
F16	SCF+LP(SUBJ+OBJ)	53.4	36.2	73.0
F17	SCF+SP	54.6	39.8	80.4

Table 2: Results for all the features for French (SPEC and K-means) and English (SPEC)

THR	Verbs	Cls	F3	F7	F13	F17
0	147	15	43.7	57.5	43.3	50.1
50	137	15	47.9	56.1	44.8	49.1
100	125	15	49.2	54.3	44.8	49.5
150	116	15	50.6	55.1	52.7	54.6
200	110	15	54.9	52.9	49.7	52.5
400	96	15	52.7	52.9	43.9	53.2
1000	71	15	51.4	54.0	44.8	54.5
2000	59	12	52.3	45.9	42.7	53.5
3000	51	12	55.7	49.0	46.8	59.2
4000	43	10	60.5	53.4	51.0	**65.4**

Table 3: The effect of verb frequency

tive feature. The only exception is the LP feature which performed better than CO in English.

7.2 Qualitative Evaluation

We conducted qualitative analysis of the clusters for French: those created using SPEC with F17 and F3. Verbs in the gold standard classes 29.2, 36.1, 37.3, 37.7 and 47.3 (Table 1) performed particularly well, with the majority of member verbs found in the same cluster. These verbs are ideal for clustering because they have distinctive syntactic-semantic characteristics. For example, verbs in 29.2 CHARACTERIZE class (e.g. *concevoir, considérer, dépeindre*) not only have a very specific meaning but they also take high frequency SCFs involving the preposition *comme* (Eng. *as*) which is not typical to many other classes. Interestingly, Levin classes 29.2, 36.1, 37.3, and 37.7 were among the best performing classes also in the supervised verb classification experiment of Sun et al. (2008) because these classes have distinctive characteristics also in English.

The benefit of sophisticated features which integrate also semantic (SP) information (F17) is particularly evident for classes with non-distinctive syntactic characteristics. For example, the intransitive verbs in 43.1 LIGHT EMISSION class (e.g. *briller, étinceler, flamboyer*) are difficult to cluster based on syntax only, but semantic features work because the verbs pose strong SPs on their subjects (entities capable of light emission). In the experiment of Sun et al. (2008), 43.1 was the worst performing class, possibly because no semantic features were used in the experiment.

The most frequent source of error is syntactic idiosyncracy. This is particularly evident for classes 10.1 REMOVE and 45.4 CHANGE OF STATE. Although verbs in these classes can take similar SCFs and alternations, only some of them are frequent in data. For example, the SCF *ôter X à Y* is frequent for verbs in 10.1, but not *ôter X de Y*. Although class 10.1 did not suffer from this problem in the English experiment of Sun et al. (2008), class 45.4 did. Class 45.4 performs particularly bad in French also because its member verbs are low in frequency.

Some errors are due to polysemy, caused partly by the fact that the French version of the gold standard was not controlled for this factor. Some verbs have their predominant senses in classes which are missing in the gold standard, e.g. the most frequent sense of *retenir* is *memorize*, not *keep* as in the gold standard class 13.5.1. GET.

Finally, some errors are not true errors but demonstrate the capability of clustering to learn novel information. For example, the CHANGE OF STATE class 45.4 includes many antonyms (e.g. *weaken* vs. *strenghten*). Clustering (using F17) separates these antonyms, so that verbs *adoucir, atténuer* and *tempérer* appear in one cluster and *consolider* and *renforcer* in another. Although these verbs share the same alternations, their SPs are different. The opposite effect can be observed when clustering maps together classes

available for a verb, CO outperforms all the other features in French, compensating for data sparseness.

which are semantically and syntactically related (e.g. 36.1 CORRESPOND and 37.7 SPEAK). Such classes are distinct in Levin and VerbNet, although should ideally be related. Cases such as these show the potential of clustering in discovering novel valuable information in data.

8 Discussion and Conclusion

When sufficient corpus data is available, there is a strong correlation between the types of features which perform the best in English and French. When the best features are used, many individual Levin classes have similar performance in the two languages. Due to differences in data sets direct comparison of performance figures for English and French is not possible. When considering the general level of performance, our best performance for French (65.4 F) is lower than the best performance for English in the experiment of Sun and Korhonen (2009). However, it does compare favourably to the performance of other state-of-the-art (even supervised) English systems (Joanis et al., 2008; Li and Brew, 2008; Ó Séaghdha and Copestake, 2008; Vlachos et al., 2009). This is impressive considering that we experimented with a fully unsupervised approach originally developed for another language.

When aiming to improve performance further, employing larger data is critical. Most recent experiments on English have employed bigger data sets, and unlike us, some of them have only considered the predominant senses of medium-high frequency verbs. As seen in section 7.1, such differences in data can have significant impact on performance. However, parser and feature extraction performance can also play a big role in overall accuracy, and should therefore be investigated further (Sun and Korhonen, 2009). The relatively low performance of basic LP features in French suggests that at least some of the current errors are due to parsing. Future research should investigate the source of error at different stages of processing. In addition, it would be interesting to investigate whether language-specific tuning (e.g. using language specific features such as auxiliary classes) can further improve performance on French.

Earlier works most closely related to ours are those of Merlo et al. (2002) and Ferrer (2004). Our results contrast with those of Ferrer who showed that a clustering approach does not transfer well from English to Spanish. However, she used basic SCF and named entity features only, and a clustering algorithm less suitable for high dimensional data. Like us, Merlo et al. (2002) created a gold standard by translating Levin classes to another language (Italian). They also applied a method developed for English to Italian, and reported good overall performance using features developed for English. Although the experiment was small (focussing on three classes and a few features only) and involved supervised classification, the results agree with ours.

These experiments support the linguistic hypothesis that Levin style classification can be cross-linguistically applicable. A clustering technique such as the one presented here could be used as a tool for investigating whether classifications are similar across a wider range of more diverse languages. From the NLP perspective, the fact that an unsupervised technique developed for one language can be applied to another language without the need for substantial tuning means that automatic techniques could be used to hypothesise useful Levin style classes for further languages. This, in turn, could facilitate the creation of multilingual VerbNets in the future.

9 Acknowledgement

Our work was funded by the Royal Society University Research Fellowship (AK), the Dorothy Hodgkin Postgraduate Award (LS), the EPSRC grants EP/F030061/1 and EP/G051070/1 (UK) and the EU FP7 project 'PANACEA'.

References

Omri Abend, Roi Reichart, and Ari Rappoport. A supervised algorithm for verb disambiguation into VerbNet classes. In *Proc. of COLING*, pages 9–16, 2008.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The Berkeley FrameNet Project. In *COLING-ACL*, pages 86–90, 1998.

Didier Bourigault, Marie-Paule Jacques, Cécile Fabre, Cécile Frérot, and Sylwia Ozdowska. Syntex, analyseur syntaxique de corpus. In *Actes des*

12èmes journées sur le Traitement Automatique des Langues Naturelles, 2005.

Chris Brew and Sabine Schulte im Walde. Spectral clustering for German verbs. In *Proc. of EMNLP*, pages 117–124, 2002.

Jinxiu Chen, Dong-Hong Ji, Chew Lim Tan, and Zheng-Yu Niu. Unsupervised relation disambiguation using spectral clustering. In *Proc. of COLING/ACL*, pages 89–96, 2006.

Hoa Trang Dang. *Investigations into the Role of Lexical Semantics in Word Sense Disambiguation*. PhD thesis, CIS, University of Pennsylvania, 2004.

Eva Esteve Ferrer. Towards a semantic classification of Spanish verbs based on subcategorisation information. In *Proc. of ACL Student Research Workshop*, 2004.

Ralph Grishman, Catherine Macleod, and Adam Meyers. Comlex syntax: building a computational lexicon. In *Proc. of COLING*, pages 268–272, 1994.

Maurice Gross. *Méthodes en syntaxe*. Hermann, Paris, 1975.

Eduard Hovy, Mitch Marcus, Martha Palmer, L. Ramshaw, and R. Weischedel. Ontonotes: The 90% solution. In *HLT/NAACL*, 2006.

Ray Jackendoff. *Semantic Structures*. The MIT Press, Cambridge, MA, 1990.

Eric Joanis, Suzanne Stevenson, and David James. A general feature space for automatic verb classification. *Nat. Lang. Eng.*, 14(3):337–367, 2008.

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. A large-scale classification of English verbs. *Language Resources and Evaluation*, 42:21–40, 2008.

Karin Kipper-Schuler. *VerbNet: A broad-coverage, comprehensive verb lexicon*. University of Pennsylvania, PA, 2005.

Beth. Levin. English verb classes and alternations: A preliminary investigation. *Chicago, IL*, 1993.

Jianguo Li and Chris Brew. Which Are the Best Features for Automatic Verb Classification. In *Proc. of ACL*, pages 434–442, 2008.

Marina. Meila. The multicut lemma. Technical report, University of Washington, 2001.

Marina Meila and Jianbo Shi. A random walks view of spectral segmentation. In *AISTATS*, 2001.

Paola Merlo, Suzanne Stevenson, Vivian Tsang, and Gianluca Allaria. A multilingual paradigm for automatic verb classification. In *Proc. of ACL*, 2002.

Cédric Messiant. ASSCI : A subcategorization frames acquisition system for French. In *Proc. of ACL Student Research Workshop*, pages 55–60, 2008.

Cédric Messiant, Thierry Poibeau, and Anna Korhonen. LexSchem: a Large Subcategorization Lexicon for French Verbs. In *Proc. of LREC*, 2008.

George A. Miller. WordNet: a lexical database for English. *Communications of the ACM*, 1995.

Diarmuid Ó Séaghdha and Ann Copestake. Semantic classification with distributional kernels. In *Proc. of COLING*, pages 649–656, 2008.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 3(1):71–106, 2005.

Patrick Saint-Dizier. Verb Semantic Classes Based on 'alternations' and WordNet-like criteria . In P. Saint-Dizier, editor, *Predicative Forms in Natural language and lexical Knowledge Bases* , pages 247–279. Kluwer Academic, 1998.

Sabine Schulte im Walde. Experiments on the Automatic Induction of German Semantic Verb Classes. *Computational Linguistics*, 2006.

Lei Shi and Rada Mihalcea. Putting pieces together: Combining FrameNet, VerbNet and WordNet for robust semantic parsing. In *Proc. of CICLing*, pages 100–111, 2005.

Lin Sun and Anna Korhonen. Improving verb clustering with automatically acquired selectional preferences. In *Proc. of EMNLP*, pages 638–647, 2009.

Lin Sun, Anna Korhonen, and Yuval Krymolowski. Verb class discovery from rich syntactic data. *LNCS*, 4919:16, 2008.

Yoshimi Suzuki and Fumiyo Fukumoto. Classifying Japanese Polysemous Verbs based on Fuzzy C-means Clustering. In *Proc. of TextGraphs-4*, pages 32–40, 2009.

Robert Swier and Suzanne Stevenson. Unsupervised semantic role labelling. In *Proc. of EMNLP*, 2004.

Gloria Vázquez, Ana Fernández, Irene Castellón, and M. Antonia Martí. Clasificación verbal: Alternancias de diátesis. In *Quaderns de Sintagma*. Universitat de Lleida, 2000.

Deepak Verma and Marina Meila. A comparison of spectral clustering algorithms. Technical report, Department of CSE University of Washington Seattle, 2005.

Andreas Vlachos, Anna Korhonen, and Zoubin Ghahramani. Unsupervised and Constrained Dirichlet Process Mixture Models for Verb Clustering. In *Proc. of the Workshop on on GEMS*, pages 74–82, 2009.

Ulrike von Luxburg. A tutorial on spectral clustering. *STAT COMPUT*, 17:395 – 416, 2007.

Piek Vossen. *EuroWordNet: A Multilingual Database with Lexical Semantic Networks*. Kluwer Academic Publishers, Dordrecht, 1998.

Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. *NIPS*, 17(1601-1608):16, 2004.

Semi-supervised dependency parsing using generalized tri-training

Anders Søgaard and Christian Rishøj
Center for Language Technology
University of Copenhagen
{soegaard|crjensen}@hum.ku.dk

Abstract

Martins et al. (2008) presented what to the best of our knowledge still ranks as the best overall result on the CONLL-X Shared Task datasets. The paper shows how triads of stacked dependency parsers described in Martins et al. (2008) can label unlabeled data for each other in a way similar to co-training and produce end parsers that are significantly better than any of the stacked input parsers. We evaluate our system on five datasets from the CONLL-X Shared Task and obtain 10–20% error reductions, incl. the best reported results on four of them. We compare our approach to other semi-supervised learning algorithms.

1 Introduction

Semi-supervised learning of structured variables is a difficult problem that has received considerable attention recently, but most results have been negative (Abney, 2008). This paper uses stacked learning (Wolpert, 1992) to reduce structured variables, i.e. dependency graphs, to multinomial variables, i.e. attachment and labeling decisions, which are easier to manage in semi-supervised learning scenarios, and which can later be combined into dependency trees using parsing algorithms for arc-factored dependency parsing. Our approach thus combines ensemble-based methods and semi-supervised learning.

Ensemble-based methods such as stacked learning are used to reduce the instability of classifiers, to average out their errors and to combine the strengths of diverse learning algorithms. Ensemble-based methods have attracted a lot of attention in dependency parsing recently (Sagae and Lavie, 2006; Hall et al., 2007; Nivre and McDonald, 2008; Martins et al., 2008; Fishel and Nivre, 2009; Surdeanu and Manning, 2010). Nivre and McDonald (2008) were first to introduce stacking in the context of dependency parsing.

Semi-supervised learning is typically motivated by data sparseness. For many classification tasks in natural language processing, labeled data can be in short supply but unlabeled data is more readily available. Semi-supervised methods exploit unlabeled data in addition to labeled data to improve performance on classification tasks. If the predictions of a learner l on unlabeled data are used to improve a learner l' in semi-supervised learning, the robustness of learning will depend on the stability of l. Combining ensemble-based and semi-supervised methods may thus lead to more robust semi-supervised learning.

Ensemble-based and semi-supervised methods are some of the areas that receive most attention in machine learning today, but relatively little attention has been given to *combining* these methods (Zhou, 2009). Semi-supervised learning algorithms can be categorized with respect to the number of views, i.e. the number of feature sets, and the number of learners used to inform each other (Hady and Schwenker, 2008). Self-training and expectation maximization are perhaps the best known semi-supervised learning algorithms (Abney, 2008). They are both single-view and single-learner algorithms. Since there is thus only a single perspective on data,

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1065–1073,
Beijing, August 2010

selecting unlabeled data points with predictions is a difficult task. There is an imminent danger that the learner amplifies its previous mistakes, and while several techniques such as balancing and throttling have been developed to avoid such caveats, using single-view and single-learner algorithms often requires both caution and experience with the modeling task at hand.

Algorithms with multiple views on data are known to be more robust. This insight led to the development of co-training (Blum and Mitchell, 1998), a two-view method where views inform each other, but it also paved the way for the integration of ensemble-based and semi-supervised methods, i.e. for methods with multiple learners. It was mentioned that relatively little work has been devoted to this topic, but there are notable exceptions:

Bennett et al. (2003) generalized boosting to semi-supervised learning in a seminal paper, where the idea of iterative or recursive ensembles was also introduced. Li and Zhou (2005) introduce *tri-training*, a form of co-training that trains an ensemble of three learners on labeled data and runs them on unlabeled data. If two learners agree on their labeling of a data point, the data point is added to the labeled data of the third learner with the prediction of the first two. Didaci and Roli (2006) extend self-training and co-training to multiple learners. Li and Zhou (2007) generalize tri-training to larger ensembles of random trees. The technique is also known as co-forests. Hady and Schwenker (2008) generalize existing ensemble-based methods for semi-supervised learning scenarios; in particular they embed ensembles in a form of co-training that is shown to maintain the diversity of the ensemble over time. Milidiu and Duarte (2009) generalize boosting at start to semi-supervised learning.

This paper applies a generalization of tri-training to two classification problems, attachment and labeling. The attachment classifier's weights are used for arc-factored dependency parsing, and the labeling classifier's weights are then used to label the dependency tree delivered by the parser.

Semi-supervised dependency parsing has attracted a lot of attention recently (Koo et al., 2008; Wang et al., 2008; Suzuki et al., 2009), but there has, to the best of our knowledge, been no previous attempts to apply tri-training or related combinations of ensemble-based and semi-supervised methods to any of these tasks, except for the work of Sagae and Tsujii (2007) discussed in Sect. 2.6. However, tri-training has been applied to Chinese chunking (Chen et al., 2006), question classification (Nguyen et al., 2008) and POS tagging (Søgaard, 2010).

We compare generalized tri-training to other semi-supervised learning algorithms, incl. self-training, the original tri-training algorithm based on bootstrap samples (Li and Zhou, 2005), co-forests (Li and Zhou, 2007) and semi-supervised support vector machines (Sindhwani and Keerthi, 2006).

Sect. 2 introduces dependency parsing and stacked learning. Stacked learning is generalized to dependency parsing, and previous work is briefly surveyed. We then describe how stacked dependency parsers can be further stacked as input for two end classifiers that can be combined to produce dependency structures. These two classifiers will learn multinomial variables (attachment and labeling) from a combination of labeled data and unlabeled data using a generalization of tri-training. Sect. 3 describes our experiments. We describe the data sets, and how the unlabeled data was prepared. Sect. 4 presents our results. Sect. 5 presents an error analysis and discusses the results in light of other results in the literature, and Sect. 6 concludes the paper.

2 Background and related work

2.1 Dependency parsing

Dependency parsing models a sentence as a tree where words are vertices and grammatical functions are directed edges (dependencies). Each word thus has a single incoming edge, except one called the root of the tree. Dependency parsing is thus a structured prediction problem with trees as structured variables. Each sentence has exponentially many possible dependency trees. Our observed variables are sentences with words labeled with part-of-speech tags. The task for

each sentence is to find the dependency tree that maximizes an objective function which in our case is learned from a combination of labeled and unlabeled data.

More formally, a dependency tree for a sentence $x = w_1, \ldots, w_n$ is a tree $T = \langle \{0, 1, \ldots, n\}, A \rangle$ with $A \subseteq V \times V$ the set of dependency arcs. Each vertex corresponds to a word in the sentence, except 0 which is the root vertex, i.e. for any $i \leq n$ $\langle i, 0 \rangle \notin A$. Since a dependency tree is a tree it is acyclic. A tree is projective if every vertex has a continuous projection, i.e. if and only if for every arc $\langle i, j \rangle \in A$ and node $k \in V$, if $i < k < j$ or $j < k < i$ then there is a subset of arcs $\{\langle i, i_1 \rangle, \langle i_1, i_2 \rangle, \ldots, \langle i_{k-1}, i_k \rangle\} \in A$ such that $i_k = k$.

In this paper we use a maximum spanning tree algorithm, the so-called Chu-Liu-Edmonds algorithm (CLE) (Edmonds, 1967) to turn the predictions of our semi-supervised classifiers into a dependency tree.

2.2 Stacked learning

Stacked generalization, or simply *stacking*, was first proposed by Wolpert (1992). Stacking is an ensemble-based learning method where multiple weak classifiers are combined in a strong end classifier. The idea is to train the end classifier directly on the predictions of the input classifiers.

Say each input classifier c_i with $1 \leq i \leq n$ receives an input $\mathbf{x}$ and outputs a prediction $c_i(\mathbf{x})$. The end classifier then takes as input $\langle \mathbf{x}, c_1(\mathbf{x}), \ldots, c_n(\mathbf{x}) \rangle$ and outputs a final prediction $c_0(\langle \mathbf{x}, c_1(\mathbf{x}), \ldots, c_n(\mathbf{x}) \rangle)$. Training is done by cross-validation. In sum, stacking is training a classifier on the output of classifiers.

2.3 Stacked dependency parsing

Stacked learning can be generalized to structured prediction tasks such as dependency parsing. Architectures for stacking dependency parsers typically only use one input parser, but otherwise the intuition is the same: the input parser is used to augment the dependency structures that the end parser is trained and evaluated on.

Nivre and McDonald (2008) first showed how the MSTParser (McDonald et al., 2005) and the MaltParser (Nivre et al., 2007) could be improved by stacking each parser on the predictions of the other. Martins et al. (2008) generalized their work, considering more combinations of parsers, and stacking the end parsers on non-local features from the predictions of the input parser, e.g. siblings and grand-parents. In this work we use three stacked dependency parsers for each language: mst2 (p_1), malt/mst2 (p_2) and malt/mst1 (p_3).

The notation "malt/mst2" means that the second-order MSTParser has been stacked on the MaltParser. The capital letters refer to feature configurations. Configuration D stacks a level 1 parser on several (non-local) features of the predictions of the level 0 parser (along with the input features): the predicted edge, siblings, grand parents and predicted head of candidate modifier if predicted edge is 0. Configuration E stacks a level 1 parser on the features in configuration D and all the predicted children of the candidate head. The chosen parser configurations are those that performed best in Martins et al. (2008) across the different datasets.

2.4 Stacking stacked dependency parsing

The input features of the input classifiers in stacked learning $\mathbf{x}$ can of course be removed from the input of the end classifier. It is also possible to stack stacked classifiers. This leaves us with four strategies for recursive stacking; namely to constantly augment the feature set, with level n classifiers trained on the predictions of the classifiers at all $n-1$ lower levels with or without the input features $\mathbf{x}$, or simply to train a level n classifier on the predictions of the level $n-1$ classifiers with or without $\mathbf{x}$.

In this work we stack stacked dependency parsers by training classifiers on the output of three stacked dependency parsers and POS tags. Consequently, we use one of the features from $\mathbf{x}$. Note that we train classifiers and not parsers on this new level 2.

The reduction is done the following way: First we train a classifier on the relative distance from a word to its head to induce attachments. For example, we may obtain the following features from the predictions of our level 1 parsers:

label	p_1	p_2	p_3	POS
1	1	-1	1	NNP
0	0	0	0	VBD

In the second row all input parsers, p_{1-3} in columnsaa 2–4, agree that the verb is the root of the sentence. Column 1 tells us that this is correct. In the first row, two out of three parsers agree on attaching the noun to the verb, which again is correct. We train level 2 classifiers on feature vectors produced this way. Note that oracle performance of the ensemble is no upper bound on the accuracy of a classifier trained on level 1 predictions this way, since a classifier may learn the right decision from three wrong predictions and a POS tag.

Second we train a classifier to predict dependency relations. Our feature vectors are similar to the ones just described, but now contain dependency label predictions, e.g.:

label	p_1	p_2	p_3	POS
SBJ	SBJ	SBJ	SBJ	NN
ROOT	ROOT	ROOT	COORD	VBN

2.5 Generalized tri-training

Tri-training was originally introduced in Li and Zhou (2005). The method involves three learners that inform each other.

Let L denote the labeled data and U the unlabeled data. Assume that three classifiers c_1, c_2, c_3 have been trained on L. In the original algorithm, the three classifiers are obtained by applying the same learning algorithm to three bootstrap samples of the labeled data; but in generalized algorithms, three different learning algorithms are used. An unlabeled datapoint in U is labeled for a classifier, say c_1, if the other two classifiers agree on its label, i.e. c_2 and c_3. Two classifiers inform the third. If the two classifiers agree on a labeling, we assume there is a good chance that they are right. In the original algorithm, learning stops when the classifiers no longer change; in generalized tri-training, a fixed stopping criterion is used. The three classifiers are combined by voting. Li and Zhou (2005) show that under certain conditions the increase in classification noise rate is compensated by the amount of newly labeled data points.

The most important condition is that the three classifiers are diverse. If the three classifiers are identical, tri-training degenerates to *self-training*. As already mentioned, Li and Zhou (2005) obtain this diversity by training classifiers on bootstrap samples. In their experiments, they consider classifiers based on decision trees, BP neural networks and naïve Bayes inference.

```
for i ∈ {1..3} do
    c_i ← train_classifier(l_i, L)
end for
repeat
    for i ∈ {1..3} do
        for x ∈ U do
            L_i ← ∅
            if c_j(x) = c_k(x)(j, k ≠ i) then
                L_i ← L_i ∪ {(x, c_j(x)}
            end if
        end for
        c_i ← train_classifier(l_i, L ∪ L_i)
    end for
until stopping criterion is met
apply c_1
```

Figure 1: Generalized tri-training.

In this paper we generalize the tri-training algorithm and use three different learning algorithms rather than bootstrap samples to create diversity: a naïve Bayes algorithm (no smoothing), random forests (Breiman, 2001) (with 100 unpruned decision trees) and an algorithm that induces unpruned decision trees. The overall algorithm is sketched in Figure 1 with l_i a learning algorithm.

Our weights are those of the random forest classifier after a fixed number of rounds. The attachment classifier iterates once over the unlabeled data, while the dependency relations classifier uses three iterations. The optimal number of iterations could of course be estimated on development data instead. Given the weights for an input sentence we use CLE to find its most likely dependency tree.

2.6 Related work

This paper uses stacking rather than voting to construct ensembles, but voting has been more

widely used in dependency parsing than stacking. Voting was first introduced in dependency parsing in Zeman and Zabokrtsky (2005). Sagae and Lavie (2006) later used weighted voting and reparsing, i.e. using CLE to find the dependency tree that reflects the maximum number of votes. They also showed that binning the vote over part-of-speech tags led to further improvements. This set-up was adopted by Hall et al. (2007) in the best performing system in the CONLL 2007 Shared Task. Fishel and Nivre (2009) later experimented with binning the vote on other features with modest improvements.

Semi-supervised dependency parsing has only recently been explored, and failures have been more frequent than successes. There are, however, noteable exceptions such as Koo et al. (2008), Wang et al. (2008), Suzuki et al. (2009) and Sagae and Gordon (2009).

The semi-supervised methods employed in these experiments are very different from more traditional scenarios such as self-training and co-training. Two approaches (Koo et al., 2008; Sagae and Gordon, 2009) use clusters obtained from large amounts of unlabeled data to augment their labeled data by introducing new features, and two approaches (Wang et al., 2008; Suzuki et al., 2009) combine probability distributions obtained from labeled data with probability distributions obtained from unlabeled data.

Successes with self-training and co-training are rare, and several authors report negative results, e.g. Spreyer and Kuhn (2009). A noteable exception in constituent-based parsing is the work of McClosky et al. (2006) who show that self-training is possible if a reranker is used to inform the underlying parser.

Sagae and Tsujii (2007) participated in (and won) the CONLL 2007 Shared Task on domain adaptation. They first trained a maximum entropy-based transition-based dependency parser on the out-of-domain labeled data and an SVM-based transition-based dependency parser on the *reversed* out-of-domain labeled data. The two parsers parse the in-domain labeled data (reversed, in the case of the SVM-based parser). Identical analyses are added to the original training set. The first parser is retrained and used to parse the test data. In sum, the authors do one round of co-training with the following selection criterion: If the two parsers produce the same dependency structures for a sentence, the dependency structure is added to the labeled data. This criterion is also the selection criterion in tri-training.

3 Experiments

3.1 Data

We use five datasets from the CONLL-X Shared Task (Buchholz and Marsi, 2006).[1] Lemmas and morphological features (FEATS) are ignored, since we only add POS and CPOS tags to unlabeled data. For German and Swedish, we use 100,000 sentences from the Leipzig Corpora Collection (Biemann et al., 2007) as unlabeled data. For Danish, Dutch, and Portuguese we use 100,000 sentences from the Europarl corpus (Koehn, 2005). The data characteristics are provided in Figure 2. The unlabeled data were POS tagged using the freely available SVMTool (Gimenez and Marquez, 2004) (model 4, left-right-left).

3.2 Algorithm

Once our data has been prepared, we train the stacked dependency parsers and use them to label training data for our classifiers ($\sim$4,000 tokens), our test data and our unlabeled data. This gives us three sets of predictions for each of the three data sets. Using the features described in Sect. 2.4 we then construct data for training our two triads of classifiers (for attachment and dependency relations). The entire architecture can be depicted as in Figure 3.

We first stack three dependency parsers as described in Martins et al. (2008). We then stack three classifiers on top of these dependency parsers (and POS tags): a naïve Bayes classifier, a random forest, and a decision tree. Finally,

[1]The CONLL-X Shared Task consists of 12 datasets, but we did not have consistently tokenized unlabeled data for Arabic, Chinese, Japanese, Slovene and Turkish. Martins et al. (2008) ignore Czech. Our experiment with the Spanish dataset crashed unexpectedly. We will post results on the website as soon as possible.

		tokens	sents	tokens/sents	POSs	DEPRELs
Danish	train	94,386	5,190	18.2	24	52
	unl (Europarl)	2,422,144	100,000	24.2	-	-
	test	5,852	322	18.2	-	-
Dutch	train	195,069	13,349	14.6	13	26
	unl (Europarl)	2,336,176	100,000	23.4	-	-
	test	5,585	386	14.5	-	-
German	train	699,610	39,216	17.8	52	46
	unl (LCC)	1,763,281	100,000	17.6	-	-
	test	5,694	357	15.9	-	-
Portuguese	train	206,678	9,071	22.3	21	55
	unl (Europarl)	2,882,967	100,000	28.8	-	-
	test	5,867	288	22.8	-	-
Swedish	train	191,467	11,042	17.4	37	56
	unl (LCC)	1,727,068	100,000	17.3	-	-
	test	5,656	389	14.5	-	-

Figure 2: Characteristics of the data sets.

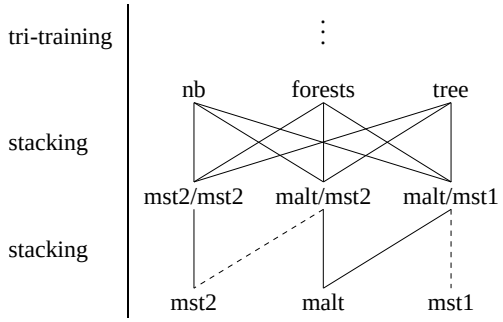


Figure 3: Tri-training stacked classifiers.

we tri-train these three stacked classifiers and for each test sentence output the weights provided by the random forest classifier. These weights are used to find the best possible dependency tree using CLE.

3.3 Baselines

The best of the stacked input parsers is of course our natural baseline.

Since we have generalized tri-training, we also compare generalized tri-training to the original tri-training algorithm based on bootstrap samples. The original tri-training algorithm is run with the same decomposition and the same features as our generalized tri-training algorithm. We use the learning algorithm originally used in Li and Zhou (2005), namely C4.5. We also compare our results to self-training (no pool, no growth rate) and co-forests (Li and Zhou, 2007). Finally, we compare our results to semi-supervised support vector machines (S3VMs) (Sindhwani and Keerthi, 2006). Since S3VMs produce binary classifiers, and one-vs.-many combination would be very time-consuming, we train a binary classifier that produces a probability that any candidate arc is correct and do greedy head selection. We optimized the feature set and included a total of seven features (head POS, dependent POS, dependent left neighbor POS, distance+direction, predictions of the three classifiers).

4 Results

Our results are presented in Figure 4. Labeled (LAS) and unlabeled attachment scores (UAS) and labeling accuracy (LA) are defined as usual and include punctuation signs unless otherwise noted. Difference (Δ) in LAS, error reduction and p-value compare our results to the best input stacked parser (malt/mst2, excerpt for Swedish).

Generalized tri-training (tri-training-CLE), i.e. using CLE to find the best well-formed dependency trees given the weights provided by our tri-trained random forest classifier, leads to highly significant improvements on *all* data sets ($p < 0.001$) with an average error reduction of 14,9%. The results for the other semi-supervised learning algorithms are presented in Figure 5. We only used 10% of the unlabeled data (10k sentences) in this experiment and only did unlabeled parsing, but it is quite evident that these learning strategies seem less promising than gen-

Danish	LAS(%)	UAS(%)	LA(%)	EM(%)	Δ LAS	err.red(%)	p-value
mst2	84.64	89.11	91.35	24.84			
malt/mst2	86.36	90.50	92.09	27.64			
malt/mst1	86.11	90.23	91.87	25.78			
tri-training-CLE	**87.76**	**92.11**	**92.87**	**27.95**	1.40	10.26	$<$0.0001
tri-training-CLE (excl. pnc.)	87.54	92.61	91.68				
CONLL-X best (excl. pnc.)	84.79	90.58	89.22				
Martins et al. (excl. pnc.)	86.79	-	-				
Dutch							
mst2	80.27	84.32	84.96	23.32			
malt/mst2	81.00	84.58	85.46	24.35			
malt/mst1	80.72	84.17	85.34	26.17			
tri-training-CLE	**83.42**	**88.18**	**87.82**	**28.00**	2.42	12.74	$<$0.0001
tri-training-CLE (excl. pnc.)	81.73	86.97	86.61				
CONLL-X best (excl. pnc.)	79.19	83.57	83.89				
Martins et al. (excl. pnc.)	81.61	-	-				
German							
mst2	87.32	89.88	93.05	35.85			
malt/mst2	88.06	90.53	93.52	40.06			
malt/mst1	88.04	90.50	93.48	38.10			
tri-training-CLE	**90.41**	**93.22**	**94.61**	**43.14**	2.35	19.68	$<$0.0001
tri-training-CLE (excl. pnc.)	90.30	93.49	93.87				
CONLL-X best (excl. pnc.)	87.34	90.38	92.11				
Martins et al. (excl. pnc.)	88.66	-	-				
Portuguese							
mst2	84.83	88.44	92.04	25.69			
malt/mst2	85.39	88.80	92.59	28.13			
malt/mst1	85.00	88.39	92.23	25.69			
tri-training-CLE	**88.03**	**91.89**	**93.54**	**29.86**	2.64	18.07	$<$0.0001
tri-training-CLE (excl. pnc.)	89.18	93.69	92.43				
CONLL-X best (excl. pnc.)	87.60	91.36	91.54				
Martins et al. (excl. pnc.)	88.46	-	-				
Swedish							
mst2	81.82	87.36	87.29	27.76			
malt/mst2	84.42	89.57	88.68	31.62			
malt/mst1	84.74	89.83	89.07	31.11			
tri-training-CLE	**86.83**	**92.04**	**90.65**	**32.65**	2.09	13.70	$<$0.0001
tri-training-CLE (excl. pnc.)	86.66	92.45	89.58				
CONLL-X best (excl. pnc.)	84.58	89.50	87.39				
Martins et al. (excl. pnc.)	85.16	-	-				
AV					2.18	14.89	

Figure 4: Results on CONLL-X datasets. Scores are **including punctuation** unless otherwise noted. Δ and p-value is difference with respect to best input parser.

UAS	malt-mst2	S3VMs	self-training	orig-tri-training	co-forests	tri-training	tri-training[full]
Danish	90.50	90.47	89.68	89.66	88.79	**90.60**	92.21
Dutch	84.58	85.34	84.06	83.83	83.97	**86.07**	88.06
German	90.53	90.15	89.83	89.92	88.47	**90.81**	93.20
Portuguese	88.80	65.64	87.60	87.62	87.06	**89.16**	91.87
Swedish	89.83	81.46	89.09	89.20	88.65	**90.22**	92.24
AV	88.80	82.61	88.05	88.05	87.44	**89.37**	91.52

Figure 5: Comparison of different semi-supervised learning algorithms (10% of unlabeled data) using 2-fold CV and no reparsing, UAS **including punctuation**.

eralized tri-training.

5 Error analysis and discussion

Error reductions are higher with dependencies to the root node and long distance dependencies than with local dependencies. The table below lists the labeled attachment F_1-scores for the five datasets binned on dependency length. The average error reduction is the same for root dependencies and long distance dependencies (length >7), but significantly lower for local dependencies. This seems to indicate that large amounts of data are necessary for the parser to recover long distance dependencies.

	root	1	2	4–7	>7
Da(F_1)	98.45	96.21	92.09	88.17	90.93
– err.red	**41.34**	10.69	13.92	15.75	21.92
Du(F_1)	83.65	94.47	88.60	82.40	81.54
– err.red	28.39	16.74	20.72	17.00	**31.88**
Ge(F_1)	97.33	96.47	94.28	92.42	93.94
– err.red	26.65	19.77	17.46	25.25	**38.97**
Po(F_1)	96.23	97.05	95.17	84.80	87.11
– err.red	22.47	19.56	24.86	22.56	**26.97**
Sw(F_1)	96.37	95.67	93.46	88.42	89.57
– err.red	**32.85**	14.10	15.04	25.97	31.50
AV err.red	**30.34**	16.17	18.40	21.31	30.25

Our results for Danish, Dutch, German and Portuguese are to the best of our knowledge the best reported results in the literature. Zhang and Chan (2009) obtain a LAS of 87.20 for Swedish with transition-based parsing based on reinforcement learning. They evaluate their system on a subset of the CONLL-X datasets and obtain their (by far) best improvement on the Swedish dataset. They speculate that "the reason might be that [long distance dependencies] are not popular in Swedish". Since our parser is particularly good at long distance dependencies, this may also explain why a supervised parser outperforms our system on this dataset. Interestingly, our unlabeled attachment score is a lot better than the one reported by Zhang and Chan (2009), namely 92.45 compared to 91.84.

Generally, our UASs are better than our LASs. Since we separate attachment and labeling out in two independent steps, improvements in UAS and improvements in LA do not necessarily lead to improvements in LAS. While our average error reduction in LAS is 14.9%, our average error reductions in UAS is 23.6%. The average error reduction in LA is 14.0%. In two-stage dependency parsers or dependency parsers with joint models, improvements in UAS are typically followed by comparable improvements in LAS.

6 Conclusion

This paper showed how the stacked dependency parsers introduced in Martins et al. (2008) can be improved by inference from unlabeled data. Briefly put, we stack three diverse classifiers on triads of stacked dependency parsers and let them label unlabeled data for each other in a co-training-like architecture. Our average error reductions in LAS over the best of our stacked input parsers is 14.9%; in UAS, it is 23.6%. The code is available at http://cst.dk/anders/tridep.html.

References

Abney, Steven. 2008. *Semi-supervised learning for computational linguistics*. Chapman & Hall.

Bennett, Kristin, Ayhan Demiriz, and Richard Maclin. 2003. Exploiting unlabeled data in ensemble methods. In *KDD*.

Biemann, Chris, G. Heyer, U. Quasthoff, and M. Richter. 2007. The Leipzig corpora collection. In *Corpus Linguistics*.

Blum, Avrim and Tom Mitchell. 1998. Combining labeled and unlabeled with-co-training. In *COLT*.

Breiman, Leo. 2001. Random forests. *Machine Learning*, 45:5–32.

Buchholz, Sabine and Erwin Marsi. 2006. CONLL-X shared task on multilingual dependency parsing. In *CONLL*.

Chen, Wenliang, Yujie Zhang, and Hitoshi Isahara. 2006. Chinese chunking with tri-training learning. In *Computer processing of oriental languages*, pages 466–473. Springer, Berlin, Germany.

Didaci, Luca and Fabio Roli. 2006. Using co-training and self-training in semi-supervised multiple classifier systems. In *SSPR& SPR*, pages 522–530. Springer, Berlin, Germany.

Edmonds, J. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71:233–240.

Fishel, Mark and Joakim Nivre. 2009. Voting and stacking in data-driven dependency parsing. In *NODALIDA*.

Gimenez, Jesus and Lluis Marquez. 2004. SVMTool: a general POS tagger generator based on support vector machines. In *LREC*.

Hady, Mohamed and Friedhelm Schwenker. 2008. Co-training by committee. *International Journal of Software and Informatics*, 2:95–124.

Hall, Johan, Jens Nilsson, Joakim Nivre, Gulsen Eryigit, Beata Megyesi, Mattias Nilsson, and Markus Saers. 2007. Single malt or blended? In *CONLL*.

Koehn, Philipp. 2005. Europarl: a parallel corpus for statistical machine translation. In *MT-Summit*.

Koo, Terry, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *ACL*.

Li, Ming and Zhi-Hua Zhou. 2005. Tri-training: exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1529–1541.

Li, Ming and Zhi-Hua Zhou. 2007. Improve computer-aided diagnosis with machine learning techniques using undiagnosed samples. *IEEE Transactions on Systems, Man and Cybernetics*, 37(6):1088–1098.

Martins, André, Dipanjan Das, Noah Smith, and Eric Xing. 2008. Stacking dependency parsers. In *EMNLP*.

McClosky, David, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *HLT-NAACL*.

McDonald, Ryan, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *HLT-EMNLP*.

Milidiu, Ruy and Julio Duarte. 2009. Improving BAS committee performance with a semi-supervised approach. In *European Symposium on Artificial Neural Networks*.

Nguyen, Tri, Le Nguyen, and Akira Shimazu. 2008. Using semi-supervised learning for question classification. *Journal of Natural Language Processing*, 15:3–21.

Nivre, Joakim and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *ACL-HLT*.

Nivre, Joakim, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser. *Natural Language Engineering*, 13(2):95–135.

Sagae, Kenji and Andrew Gordon. 2009. Clustering words by syntactic similarity improves dependency parsing of predicate-argument structures. In *IWPT*.

Sagae, Kenji and Alon Lavie. 2006. Parser combination by reparsing. In *HLT-NAACL*.

Sagae, Kenji and Jun'ichi Tsujii. 2007. Dependency parsing and domain adaptation with lr models and parser ensembles. In *EMNLP-CONLL*.

Sindhwani, Vikas and Sathiya Keerthi. 2006. Large scale semi-supervised linear SVMs. In *ACM SIGIR*.

Søgaard, Anders. 2010. Simple semi-supervised training of part-of-speech taggers. In *ACL*.

Spreyer, Kathrin and Jonas Kuhn. 2009. Data-driven dependency parsing of new languages using incomplete and noisy training data. In *CONLL*.

Surdeanu, Mihai and Christopher Manning. 2010. Ensemble models for dependency parsing: cheap and good? In *NAACL*.

Suzuki, Jun, Hideki Isozaki, Xavier Carreras, and Michael Collins. 2009. Semi-supervised convex training for dependency parsing. In *EMNLP*.

Wang, Qin, Dekang Lin, and Dale Schuurmans. 2008. Semi-supervised convex training for dependency parsing. In *ACL*.

Wolpert, David. 1992. Stacked generalization. *Neural Networks*, 5:241–259.

Zeman, Daniel and Zdeněk Žabokrtský. 2005. Improving parsing accuracy by combining diverse dependency parsers. In *IWPT*.

Zhang, Lidan and Kwok Chan. 2009. Dependency parsing with energy-based reinforcement learning. In *IWPT*.

Zhou, Zhi-Hua. 2009. When semi-supervised learning meets ensemble learning. In *MCS*.

SemanticRank: Ranking Keywords and Sentences Using Semantic Graphs

George Tsatsaronis [1] and **Iraklis Varlamis** [2] and **Kjetil Nørvåg** [1]
[1]Department of Computer and Information Science,
Norwegian University of Science and Technology
[2] Department of Informatics and Telematics, Harokopio University of Athens

Abstract

The selection of the most descriptive terms or passages from text is crucial for several tasks, such as feature extraction and summarization. In the majority of the cases, research works propose the ranking of all candidate keywords or sentences and then select the top-ranked items as features, or as a text summary respectively. Ranking is usually performed using statistical information from text (i.e., frequency of occurrence, inverse document frequency, co-occurrence information). In this paper we present *SemanticRank*, a graph-based ranking algorithm for keyword and sentence extraction from text. The algorithm constructs a *semantic graph* using implicit links, which are based on semantic relatedness between text nodes and consequently ranks nodes using different ranking algorithms. Comparative evaluation against related state of the art methods for keyword and sentence extraction shows that *SemanticRank* performs favorably in previously used data sets.

1 Introduction

Graph based ranking algorithms can be very helpful when searching for important pages in the World Wide Web, members in a social network, or authors in a publication database. Such algorithms capitalize on the existence of explicit links (e.g., hyperlinks, citations) between the graph vertices. In the case of fla text collections, neither links nor citations exist, so the need to devise implicit edges between text keywords or sentences arises. One feasible solution is to exploit the contextual information of terms and create semantic graphs from text based on content similarity. However, conceptual analysis of text has a strong potential in this direction, since it reveals latent similarities between text segments that discuss the same subject but with different terminology. In this direction, we introduce *SemanticRank*, a new algorithm for ranking text segments. *SemanticRank* comprises two steps: (a) creation of semantic graphs from text using both semantic and statistical information, and (b) application of a graph-based ranking algorithm that exploits the edges' weights.

The key contributions of this work are: (1) a novel method for the construction and weighting of the semantic graph, which contains text segments (terms or sentences) as nodes and weighted edges that capture the semantic relatedness (i.e., relatedness in meaning) between nodes but also consider statistical information, (2) the modular design of the method, which allows any graph-based ranking algorithm to be employed, and (3) thorough experimental evaluation of *SemanticRank* in the keyword extraction and text summarization tasks, and evaluation of several alternatives for the graph-based ranking component.

The paper is organized as follows: Section 2 discusses related work. Section 3 presents the preliminaries of *SemanticRank*: the semantic relatedness measure, the graph creation process and the ranking algorithm alternatives. Section 4 provides the details of our method. Section 5 presents the experimental evaluation of *SemanticRank* in two different tasks: keyword extraction and text summarization. Finally, Section 6 concludes and provides pointers to future work.

2 Related Work

This work addresses the problem of extracting the most representative keywords and sentences from text as a means of text summarization. More specificall , *SemanticRank* capitalizes on the creation of term and sentence graphs from text and on graph-based ranking algorithms in order to support the following tasks: (a) keyword extraction from text, which is performed by selecting the top-ranked terms as the most representative ones, and (b) text summarization, which is done by selecting the top ranked sentences as the most representative ones. For this reason, a survey of research works in keyword extraction and text summarization, with emphasis on graph-based approaches is necessary to understand the requirements for these tasks, to locate benchmark data sets, and state of the art graph-based approaches for the comparative evaluation.

2.1 Keyword Extraction

Keyword extraction is an important task in document indexing, and strongly affects the performance of re-

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1074–1082,
Beijing, August 2010

trieval, classification clustering, and summarization. Most keyword extraction approaches rely on statistical measures such as term frequency (TF), inverse document frequency (IDF), and variations (Aizawa, 2003). Several works in keyword extraction construct semantic networks from text in order to capture the implicit relations between the individual candidates. Huang et al. (2006) propose the construction of one semantic network per document, and use edges that capture syntactic relations between document terms. Mihalcea and Tarau (2004) suggest a semantic network model where edges express the co-occurrence of terms in the document's sentences. Wang et al. (2007) employ the well known *PageRank* algorithm to perform word sense disambiguation (WSD) and keyword extraction from documents. The graphs that they construct are always subgraphs of the *WordNet* thesaurus[1], which results in low text coverage.

In this work we aim at the design and implementation of a keyword extraction algorithm that takes into account different aspects of text, such as statistical information and the semantic relatedness between keywords. To the best of our knowledge, the current work is the firs to propose a semantic network construction model for keyword extraction based on measures of semantic relatedness between keywords. In Section 3 we discuss the employed measure of semantic relatedness, which utilizes both *WordNet* and *Wikipedia*[2] to increase term coverage.

2.2 Text Summarization

The aim of automatic text summarization is to generate a summary of a pre-specifie length for a given input text. The *Document Understanding Conference* (DUC) [3] series have provided benchmark data sets with documents and manually generated summaries, which can be used for the evaluation of any automatic text summarizer. Research works in this area conduct automatic text summarization by selecting the most important sentences from the input texts (Steinberger and Jezek, 2009). Baseline methods are based on the observation that important sentences inside a text usually occur at its beginning. Thus, a straightforward baseline is to select the firs k sentences as a summary of the text, and setting k in such a way that does not violate the summary length restriction.

Another important class of automated text summarization methods is that of cohesion-based methods. Such methods assume that the important sentences or paragraphs of a given text document are the most connected entities in more or less elaborate semantic structures inside the text. In this direction, the methods construct a graph for each text document, with the vertices being the document sentences, and attempt to determine the most connected vertices in each graph. These methods are further classifie depending on how the graph's edges are constructed, for example using word co-occurrences (Salton et al., 1997), local salience and grammatical relations (Boguarev and Kennedy, 1997), co-reference (Baldwin and Morton, 1998), and combinations of the aforementioned (Mani and Bloedorn, 1998).

More recently, some cohesion-based methods have attempted to capture the semantic similarity of sentences inside a text document, and rank sentences in the constructed semantic graph. For example in the method of Mihalcea and Tarau (2004), the graph contains a vertex for each sentence of the given text document, and weighted edges between sentence. The weights represent the semantic similarity between sentences and are actually the contextual overlap between the sentences' terms. The *PageRank* algorithm is then applied to rank sentences in each of the constructed semantic graphs. In the method of Litvak and Last (2008), the vertices are again the sentences of the given text document and the edges represent the syntactic relations between them. Finally, the *HITS* algorithm is applied on the graph for ranking the sentences.

The conclusion from the literature review is that modern trends in graph-based approaches focus on novel methodologies for weighting edges and constructing semantic graphs, and employ standard techniques for ranking vertices, such as *PageRank*, *HITS*, or variations. Another important findin is that the potentiality of creating the edges between the vertices based on measures of semantic relatedness among the respective nodes is unexploited so far, and this is the core of the current work. Thus, the main difference between *SemanticRank* and the aforementioned approaches is that our edge weighting method employs a measure of semantic relatedness between sentences, that is based on *WordNet* and *Wikipedia*. The motivation behind such a perspective is that such semantic graphs would capture the similarity in meaning among the graph vertices, which was neglected by previous approaches. Finally, regarding the data sets used for evaluation, most works in text summarization use past *DUC* data, whereas in the case of keyword extraction a subset from the *Inspec* bibliographic database has been used in several cases in the past (Mihalcea and Tarau, 2004; Hulth, 2003).

[1] http://wordnet.princeton.edu/

[2] http://wikipedia.org

[3] http://duc.nist.gov/. Recently it has been renamed to *Text Understanding Conference* (TAC).

3 Terminology and Preliminaries

A graph-based method for ranking keywords or sentences by constructing semantic graphs comprises two steps: (a) the creation of the semantic graph, with keywords or sentences as vertices, and edges constructed based on a semantic similarity measure between vertices (c.f. Section 3.2), and (b) the adaptation of a new or existent ranking algorithm which analyzes the graph structure and ranks the nodes. Section 3.1 introduces the used terminology. In Section 3.2 we explain how the firs step is done by *SemanticRank*, and in Section 3.3 we present several alternatives for the second step.

3.1 Terminology

In the following we denote with $T(t_i, t_j)$ a pair of terms that occur in text document T. We also assume that T is a member of a document collection D given as input to our method. O represents the used knowledge-base (e.g., thesaurus, dictionary); in our case we are using two such knowledge-bases, namely *WordNet* and *Wikipedia*. With $SR_O(t_i, t_j)$ we denote the semantic relatedness between terms t_i and t_j using O for its computation, and $SRS(A, B)$ represents the semantic relatedness between text segments A and B (e.g., documents, sentences).

Concerning *WordNet*, S_i (S_j) represents the set of the different meanings (senses) with which t_i (t_j) may appear in O. P_{ij} denotes the set of paths connecting senses in S_i with senses in S_j, as these may be found using O. P_{ij}^k represents one such path in the set of paths P_{ij}, namely the k_{th} path. S_{ij} stands for the set of all possible sense pairs between the set of senses S_i and the set of senses S_j. Thus, $|S_{ij}| = |S_i| * |S_j|$. Respectively, S_{ij}^m stands for one such combination, namely the m_{th} combination.

With regards to Wikipedia, W refers to all the *Wikipedia* articles. With a_i we denote the *Wikipedia* article for term t_i. $In(a_i)$ is the set of *Wikipedia* articles that contain at least one link to a_i.

Finally, if d_i is the i_{th} document of D and t_a a term in d_i, then we denote with $TF\text{-}IDF(t_a, d_i) = \frac{Count(t_a,d_i)}{|d_i|} \cdot \log_2 \frac{Count(t_a,D)+1}{|D|}$ the *TF-IDF* weight of t_a in d_i, where $|d_i|$ is the number of term occurrences in d_i, $|D|$ is the number of documents in D, $Count(t_a, d_i)$ the number of occurrences of t_a in d_i, and $Count(t_a, D)$ the number of documents in D that contain t_a.

3.2 Creating Semantic Graphs from Text

A huge volume of literature has been created on how to construct semantic graphs addressing various applications, such as word sense disambiguation (Agirre and Soroa, 2009), keyword and sentence extraction (Mihalcea and Tarau, 2004; Litvak and Last, 2008; Yeh et al., 2008), and computation of semantic relatedness or similarity between terms (Gabrilovich and Markovitch, 2007; Budanitsky and Hirst, 2006; Milne and Witten, 2008).

In this work we adopt a semantic graph construction method which is able to capture the semantic relatedness between terms, as well as text segments. For our purposes we adopt *Omiotis*, the measure proposed by Tsatsaronis et al. (2010) in order to construct and weigh the edges of the semantic graph. *Omiotis* is a knowledge-based measure of semantic relatedness that may capture the semantic relatedness between both keywords and text segments (e.g., sentences, documents), allowing us to construct both semantic keyword graphs for keyword extraction, and semantic sentence graphs for sentence extraction and summarization. Our selection also lies in the fact that *Omiotis* has been shown to perform very well compared to other known measure of semantic relatedness or similarity in tasks such as term-to-term similarity (Tsatsaronis et al., 2010; Budanitsky and Hirst, 2006). However, since *Omiotis* relies solely in *WordNet*, we enhance the coverage of *SemanticRank* by complementing the edge weighting with an additional *Wikipedia*-based measure, namely the measure proposed by Milne and Witten (*MLN*) (2008). In Section 3.2.1 we explain how these two measures are combined in order to compute the semantic relatedness between terms, and in Section 3.2.2 we explain how semantic relatedness is captured between sentences.

3.2.1 Semantic Relatedness Between Terms

The measure presented in (Tsatsaronis et al., 2010) define the semantic relatedness between a pair of terms as shown in Equation 1, where the knowledge-base O is *WordNet* (*WN*).

$$SR_{WN}(t_i,t_j)=\max_m\{\max_k\{SCM(S_{ij}^m,P_{ij}^k)\cdot SPE(S_{ij}^m,P_{ij}^k)\}\} \quad (1)$$

where *SCM* and *SPE* are called *Semantic Compactness* and *Semantic Path Elaboration* respectively. Their product measures the weight of the path connecting the two senses in S_{ij}^m, taking into account: the path length, the type of the semantic edges comprising it, and the depth of the intermediate nodes in the *WN* senses hierarchy. The semantic relatedness between two terms t_i, t_j, when $t_i \in WN$ and $t_j \notin WN$, or vice versa, is considered 0. The intuition behind Equation 1 is that the semantic relatedness between two terms should be computed based on the *highest value* path connecting any pair of senses of the two terms. The computation of the *value* takes into account in tandem all of the aforementioned factors.

In order to enhance the coverage of the measure in Equation 1, we combine it with the *WLM Wikipedia*-

based measure of Milne and Witten (2008), which is a low-cost solution for measuring relatedness between terms using the *Wikipedia* articles and link structure as a knowledge base. The semantic relatedness between two terms t_i and t_j according to *WLM* is define as shown in Equation 2. The intuition behind this formula is that the semantic similarity between two terms becomes higher, as the number of articles pointing to both respective *Wikipedia* articles increases (i.e., as the percentage of the articles linking to both pages compared to the number of articles linking to either of them increases).

$$SR_{Wiki}(t_i,t_j)=\frac{log(\max\{|In(a_i)|,|In(a_j)|\})-log(|In(a_i)\cap In(a_j)|)}{log(|W|)-log(\min\{|In(a_i)|,|In(a_j)|\})} \quad (2)$$

We combine the two measures in a single measure $SRT(t_i,t_j)$, as shown in Equation 3. The reason we prioritize $SR_{WN}(t_i,t_j)$ from $SR_{Wiki}(t_i,t_j)$, when both terms exist in *WN*, is because the former measure has shown much better performance in capturing the semantic relatedness between terms.

$$SRT(t_i,t_j)=\begin{cases}1, & t_i=t_j\\ SR_{WN}(t_i,t_j), & \text{if } t_i,t_j\in \textit{WordNet}\\ SR_{Wiki}(t_i,t_j), & \text{if } t_i,t_j\in \textit{Wikipedia}\\ 0, & \text{otherwise}\end{cases} \quad (3)$$

3.2.2 Semantic Relatedness Between Texts

To quantify the semantic relatedness for a pair of text segments, we build upon the *SRT* measure, but also take into account the statistical importance of the terms occurring in the respective texts. Given two text segments A and B, and two terms $t_a \in A$ and $t_b \in B$, a measure that combines the statistical importance of t_a and t_b, according to (Tsatsaronis et al., 2010), is the harmonic mean of their *TF-IDF* weights. We denote this quantity as λ_{t_a,t_b}. Then for each term $t_a \in A$, we search for the corresponding term $t_b \in B$, which we symbolize with b_*, that maximizes the product of their combined statistical importance and semantic similarity. In our case, b_* is found by Equation 4. Similarly we can fin for each $t_b \in B$ the corresponding a_*.

$$b_* = \arg\max_{t_b\in B}\{\lambda_{t_a,t_b}\cdot SRT(t_a,t_b)\} \quad (4)$$

After findin the set of all b_* and a_* terms, the semantic relatedness between the two texts A and B is computed as shown in Equation 5.

$$SRS(A,B)=\frac{\theta(A,B)+\theta(B,A)}{2} \quad (5)$$

where $\theta(A,B)=\frac{1}{|A|}\sum_{t_a\in A}\lambda_{t_a,b_*}\cdot SRT(t_a,b_*)$, and $\theta(B,A)$ can be computed respectively. The measure in Equation 5 is the measure used by *SemanticRank* to construct the edges between sentence vertices in the case of the semantic sentence graphs for text summarization. Regarding which sense of each term is used for the computation of its semantic relatedness with any other term, the senses that maximize the measure in Equation 3 are picked in each case.

3.3 Ranking Nodes in Semantic Graphs

For the purposes of our experimentation we will be evaluating *SemanticRank* with variations of the known *PageRank* and *HITS* algorithms. Some of those variations are applied for the firs time in the framework of ranking nodes in semantic graphs. However, as will be explained in Section 4, in *SemanticRank*, any available vertex ranking methodology can be used instead.

The original versions of *PageRank* and *HITS* rely on the *"rich get richer"* model, which is based on explicit links and ignores edges weights. More specifically, HITS prioritizes good hubs and authorities, whereas PageRank uses a dampening factor (β) in order to avoid clique attacks and promote the centrality of nodes. However, in the case of graphs with implicitly devised links, like in semantic graphs, the edges carry weights, which must be taken into account. In this direction, we employ a modifie version of the original *PageRank* algorithm, firs introduced by Mihalcea and Tarau (2004). The modifie *PageRank* is shown in Equation 6.

$$WPR(i)=(1-\beta)+\beta\cdot\sum_{j\in IN(i)}\frac{w_{ij}\cdot WPR(j)}{\sum_{k\in OUT(j)}w_{jk}} \quad (6)$$

where i,j,k represent vertices, $IN(i)$ and $OUT(j)$ are the sets of *inlink* nodes of i and *outlink* nodes of j respectively, and w_{ij} is the weight of the edge between nodes i and j. In the case of semantic graphs constructed for keyword extraction, nodes are terms, and, thus, $w_{ij} = SRT(t_i,t_j)$. In the case of semantic graphs constructed for text summarization, i and j are sentences, and, thus, $w_{ij} = SRS(i,j)$.

Similarly to the modificatio shown in Equation 6 for *PageRank*, we can defin a weighted version of *HITS*. The respective *authority* and *hub* scores are shown in Equations 7 and 8.

$$authority(i)=\sum_{j\in In(i)}w_{i,j}\cdot hub(j) \quad (7)$$

$$hub(i)=\sum_{j\in Out(i)}w_{i,j}\cdot authority(j) \quad (8)$$

The aforementioned modification have been already applied in the past in the case of semantic

graphs, with application to keyword extraction and text summarization (Mihalcea and Tarau, 2004; Mihalcea, 2004), although using different semantic graphs. For the extraction of the most important nodes, the modifie *PageRank* version is used to rank the nodes according to their fina *PageRank* values, and the modifie *HITS* to rank nodes according to their fina *authority* values. In this work, we also consider and evaluate two additional modification of *PageRank* in order to rank vertices in the case of the semantic keyword graphs. The firs modification that we call *Averaged PageRank Weighting* (*APW*) is presented in Equation 9, and is used after the weighted *PageRank* of Equation 6 has executed. The intuition behind *APW* is that each vertex t_i in the case of the keyword semantic graphs, has a known importance based on its frequency of occurrence (*TF-IDF* weight) inside the given document collection D. Thus, *APW* considers both the importance of vertex t_i inside its semantic graph, and inside its document collection.

$$APW(t_i) = \frac{1}{2}\left(\frac{WPR(t_i)}{WPR_{max}} + \frac{TF\text{-}IDF(t_i, d_j)}{TF\text{-}IDF_{max}}\right) \quad (9)$$

where d_j the specifi document from which the semantic keyword graph is created, WPR_{max} is the maximum *PageRank* score found in this graph, and $TF\text{-}IDF_{max}$ is the maximum *TF-IDF* weight found in document d_j.

The second *PageRank* modificatio that is employed for the firs time in the case of semantic keyword graphs is the *priors biased PageRank* (*P-PR*) discussed in (White and Smyth, 2003). The idea is very similar to the works in (Haveliwala, 2002) and (Agirre and Soroa, 2009), and pertain to ranking the nodes in the graph, with regards to a given set of nodes called *priors*. In short, while *PageRank* provides a global ranking of the nodes in the graph, *P-PR* provides a ranking of the nodes with regards to the set of the given *prior* nodes. This is expressed in Equation 10. The only difference with equation 6 is that each node i has its own "*random jump*" probability to the *prior* nodes. Thus, for each node i, *P-PR* has a β_i, which expresses how often we may jump back to the set of the *prior* nodes from node i. The intuition behind *priors* is that certain nodes in the graph are favored against other. In a keyword extraction task the *priors* set may contain the keywords appearing in the document's title.

$$P\text{-}PR(i) = (1-\beta_i) + \beta_i \cdot \sum_{j \in IN(i)} \frac{w_{ij} \cdot P\text{-}PR(j)}{\sum_{k \in OUT(j)} w_{jk}} \quad (10)$$

4 SemanticRank

In this section we present *SemanticRank* (illustrated in Algorithm 1), our algorithm for ranking terms and sentences based on their semantic relatedness. The firs step of *SemanticRank* is the semantic graph creation. In the case of semantic keyword graphs, and given a document d_j which belongs in a document collection D, as a preprocessing step, the algorithm detects all $n-$grams of size up to 5 words using a dictionary look-up (i.e., both *WordNet* and *Wikipedia*), and a sliding window, in order to identify candidate keywords, which may be essentially composite terms. The resulting set of terms (i.e., can be terms of 1 to 5 words), which we denote as K_{d_j}, is used for the creation of a graph G with the vertices being all the distinct terms $t_i \in K_{d_j}$. As edge weights w_{ij} *SemanticRank* uses $SRT(t_i, t_j)$ which captures the semantic relatedness between terms t_i and t_j. However, ideally we would also like to incorporate in w_{ij} the statistical information of terms t_i, t_j that we have from

Algorithm 1 SemanticRank(D,*Mode*)

INPUT: A text document collection D, and a *Mode* flag
OUTPUT: A ranking R of the semantic graph nodes for every document $d_j \in D$.
Execute(D,Mode)
if *Mode* is *Keywords* **then**
  Identify composite terms of length up to 5 words
end if
Compute and index *TF-IDF* values for all terms
for all $d_j \in D$ **do**
  G: An initially empty graph
  G = Construct_Semantic_Graph(d_j,*Mode*)
  R = Rank_Nodes(G)
end for
Construct_Semantic_Graph(d_j,Mode)
G: an initially empty graph
if *Mode* is *Keywords* **then**
  Initialize G with K_{d_j}
else
  Initialize G with Sen_{d_j}
end if
for all pairs of vertices (v_i, v_j) **do**
  if *Mode* is *Keywords* **then**
    $w_{i,j} = w_{j,i} = \lambda_{v_i,v_j} \cdot SRT(v_i, v_j)$
  else
    $w_{i,j} = w_{j,i} = SRS(v_i, v_j)$
  end if
end for
RETURN G
Rank_Nodes(G)
Execute Weighted PageRank in G
R = Rank vertices of G in descending order of PageRank values
RETURN R with their PageRank values

their frequency of occurrence inside d_j and D. Thus, Equation 11 shows this combination, and it is the formula according to which *SemanticRank* computes the edge weights w_{ij} in the case of the semantic keyword graphs. In the case of semantic sentence graphs creation, *SemanticRank* initializes G with all the distinct sentences Sen_i in d_j as vertices, and it uses Equation 5 to compute the weights between every pair of vertices (i.e., between every pair of sentences). In Algorithm 1 we denote the set of distinct sentences in d_j with Sen_{d_j}.

$$w_{ij} = \lambda_{t_i,t_j} \cdot SRT(t_i, t_j) \tag{11}$$

In both cases, for the given document d_j, and after the creation of the semantic graph, nodes may be ranked according to the values produced by applying either Equation 6, or Equations 7 and 8. For the case of semantic keyword graphs, the top-k ranked nodes are selected as the most important keywords of d_j. For the case of semantic sentence graphs, the top-k ranked nodes are selected as the set of sentences, put together to constitute the automatically generated summary of d_j. In Algorithm 1 we may substitute line 26 with any of the ranking options discussed in Section 3.3. An analogy can be also drawn with PageRank's *random surfer model*, where a user browses the Web by following links from any given Web page. In the context of text modelling, *SemanticRank* implements what we refer to as *text surfing*, which relates to the concept of text cohesion (Halliday and Hasan, 1976), i.e., from a certain concept in a text, we are likely to *follow* "links" to related concepts, meaning concepts that have lexical or semantic relation to the current concept.

5 Experimental Evaluation

The experimental evaluation is performed in two tasks: (a) keyword extraction, and (b) text summarization. In both cases we create a semantic graph for each document and we rank the nodes accordingly, using Algorithm 1. For our evaluation we use all the ranking algorithm alternatives described in Section 3.3, and compare results with state of the art approaches that use the same ranking algorithms but different graph creation and edge weighting approaches. The various tested ranking alternatives are: weighted *SemanticRank* (*Sem*) using *PageRank* (*WPR*), and *HITS* (*WHITS*), and unweighted *SemanticRank* (*USem*) using the original versions of *PageRank* (*UPR*) and *HITS* (*UHITS*). In the case of keyword extraction we evaluate additionally the *Averaged PageRank Weighting* (*APW*) and *PageRank Priors* (*PPR*), where the *prior* nodes were set to the terms occurring in each abstract's title.

Method		P	R	F
Sem (k=5)	WPR	0.396	0.121	0.1853
	WHITS	0.348	0.088	0.14
	APW	0.556	0.185	0.278
	P-PR	**0.659**	0.226	0.337
Sem (k=10)	WPR	0.368	0.2463	0.296
	WHITS	0.335	0.138	0.195
	APW	0.498	0.331	0.398
	P-PR	0.524	0.352	0.422
Sem (k=15)	WPR	0.371	0.364	0.368
	WHITS	0.355	0.241	0.287
	APW	0.449	0.442	0.446
	P-PR	0.451	0.441	0.446
Sem (k=20)	WPR	0.376	0.466	0.417
	WHITS	0.374	0.312	0.34
	APW	0.421	**0.532**	**0.47**
	P-PR	0.418	0.514	0.46
USem (k=5)	UPR	0.057	0.046	0.048
	UHITS	0.061	0.053	0.055
USem (k=10)	UPR	0.06	0.102	0.07
	UHITS	0.06	0.108	0.072
USem (k=15)	UPR	0.052	0.116	0.069
	UHITS	0.054	0.123	0.072
USem (k=20)	UPR	0.052	0.14	0.074
	UHITS	0.053	0.151	0.076
Michalcea (2004)		0.312	0.431	0.362
Hulth (2003)		0.252	0.517	0.339

Table 1: Results of the keyword extraction task in the Inspec database.

5.1 Keyword Extraction

We applied *SemanticRank* in an automated keyword extraction task on the Inspec database[4]. The Inspec database stores abstracts of journal papers from computer science and information technology and the keyword extraction task aims in selecting the most descriptive keywords for each abstract. Each abstract has been already assigned keywords by professional indexers, which constitute the gold standards for systems' comparison. The mean number of assigned terms per abstract from the experts is 7.63. The goal is to extract as many of the keywords suggested by the professional indexers as possible for each abstract. In this data set our results are directly comparable to the works in (Mihalcea and Tarau, 2004) and (Hulth, 2003).

We evaluate *SemanticRank* (*Sem*) using varying k values ($5, 10, 15$, and 20), where k stands for the number of keywords to be extracted from each abstract. In Table 1 we report the results of macro-averaged pre-

[4]Many thanks to Anette Hulth for providing us the data set used in her keyword extraction experiments.

System		F-Measure
Sem	WPR	$0.40996(0.39067 - 0.4292)$
	WHITS	$0.3651(0.3435 - 0.38609)$
USem	UPR	$0.2951(0.2727 - 0.3195)$
	UHITS	$0.3132(0.2901 - 0.3375)$
T		$0.4131(0.3922 - 0.434)$
P		$0.4039(0.3843 - 0.4226)$
O		$0.3905(0.3663 - 0.4132)$
V		$0.3885(0.368 - 0.4085)$
Q		$0.3857(0.3616 - 0.4089)$
Baseline		$0.3549(0.3329 - 0.3756)$

Table 2: Results (F-Measure) of the single-document summarization task, (DUC 2001).

System		F-Measure
Sem	WPR	$0.4971(0.4799 - 0.5164)$
	WHITS	$0.3836(0.3815 - 0.4047)$
USem	UPR	$0.3086($ 0.297-0.32084$)$
	UHITS	$0.2851($ 0.2735-0.297$)$
TextRank		0.4904
S27		0.5011
S31		0.4914
S28		0.489
S21		0.4869
S29		0.4681
Baseline		0.4779

Table 3: Results (F-Measure) of the single-document summarization task, (DUC 2002).

cision (P), recall (R), and F-Measure (F) over all abstracts. Precision for each abstract is the number of correctly extracted keywords, divided by the number of extracted keywords, and recall differs only in the denominator (number of keywords suggested by the indexers). We also present the best reported results for the algorithms in (Mihalcea and Tarau, 2004), and (Hulth, 2003).

Results show that *SemanticRank* with weighted *PageRank* gives better F-Measure from the approaches in (Mihalcea and Tarau, 2004) and (Hulth, 2003) for $k = 15$ and $k = 20$ and always better from weighted *HITS*. *APW* and *P-PR* have higher F-Measure than *WPR*, achieving top performance (bold values), with *APW* producing the best F-Measure for $k = 20$. In this case, the difference between *APW* and *TextRank*, both in precision and recall, was found statistically signifi cant at the 0.95 confidenc level, using Fisher's exact test. In addition, we can observe that the unweighted versions of *PageRank* and *HITS* produce very poor results. This shows that our method benefit greatly from the suggested edges' weighing scheme.

5.2 Text Summarization

We evaluated *SemanticRank* in two different text summarization tasks: single-document, and multi-document summarization. As in the keyword extraction task, we evaluate both the weighted and the unweighted versions of *SemanticRank* (*Sem* and *USem*) using *WPR*, *WHITS*, *UPR*, and *UHITS* respectively. We also compare against state of the art results in the used data sets, and we report on results from related methods (i.e., *TextRank*) when possible.

5.2.1 Single Document Summarization

In the single-document summarization task we have used the data sets of the *Document Understanding Conference* (*DUC*) from the 2001 and 2002 competitions. The two data sets comprise 308 and 567 news articles respectively. For both data sets, two reference summaries per document were provided. The task for the participating systems in both competitions was to provide for each document a summary of at most 100 words. Thus, we apply *SemanticRank* by firs ranking sentences following Algorithm 1, and then by merging them, starting from the top ranked sentences, until the 100 words limit is reached. For the evaluation against the reference summaries, we are using the *ROUGE* toolkit, which is based on $N-$grams, and has been the standard evaluation methodology for the summarization task (Lin and Hovy, 2003) in all the recent DUC competitions. Since in *DUC* 2001 and *DUC* 2002 the *ROUGE* system was not the standard evaluation toolkit, we implemented the evaluation of the two tasks in *ROUGE*. The setup we adopted for ROUGE was (*Ngram(1,1)*, stemmed words and no stopwords), identical to the one adopted in (Mihalcea and Tarau, 2004).

In Table 2 we present the F-Measure values produced from *ROUGE* for *SemanticRank*, and the top 5 performing systems (participating systems *T*, *P*, *O*, *V*, and *Q*), for the 2001 data set. Similarly, Table 3 presents the results for the 2002 data set. In both cases we report the performance of a simple baseline method, that takes the firs sentences from each article, until the limit of 100 words is reached. When available, we also present the results from (Mihalcea and Tarau, 2004), and also the 0.95 confidenc intervals for the F-Measure values, as these were generated by *ROUGE*. The results in the two tables show that *SemanticRank*, when the weighted version of PageRank is used, produces very high F-Measure score. In both cases, our system ranks among the top 2 systems in the task.

System		F (R-2)	F (R-SU4)
Sem	WPR	0.093	0.133
	WHITS	0.078	0.115
USem	UPR	0.031	0.069
	UHITS	0.028	0.062
S40		0.111	0.143
S55		0.098	0.135
S45		0.096	0.132
S44		0.093	0.136
S47		0.093	0.130
Baseline		0.085	0.122

Table 4: Results of the multi-document summarization task (DUC 2007 update task).

5.2.2 Multi Document Summarization

For the multi document summarization task we used the data from the *DUC 2007 update task*. The data set consists of 250 documents organized in topics, and each topic is further divided into three clusters, for each of which gold standard summaries are provided by evaluators. In this case, the average of *ROUGE-2* and *ROUGE-SU4* scores are used for evaluation. Table 4 presents the average F-Measure values for both scores. We also report the top-5 performing systems in the respective task, as well as the performance of the generic baseline that was used in this case. As Table 4 shows, the combination of *SemanticRank* with the weighted *PageRank* produces better results than weighted *HITS* and the unweighted versions. This drop in performance compared to the results in the single-document summarization task can be partly explained by the fact that in this case the *DUC 2007 update task* allows for the system to assume previous knowledge for the document clusters B and C of each topic. In our case, we have not embedded any methodology that takes advantage of this knowledge.

Regarding the top system in Table 4, system *S40*, is the system called *GISTEXTER* (Hickl et al., 2007). *GISTEXTER* uses textual inference and textual contradiction to construct representations of knowledge encoded in a document collection. The system comprises four components: question processing, sentence retrieval, sentence ranking, and summary generation. However, for the summary generation component, a set of heuristics is used to generate the summary. In a similar approach (Amini and Usunier, 2007), where coherent text fragments are sought with regards to the initial question, the authors show that query expansion using a contextual approach may lead to fin important terms for the summary, among different related documents. Their system ranked among the top in the main task of *DUC 2007*, leading to the conclusion that for a multi-document summarization system, a contextual approach might be more efficien than *SemanticRank*.

However, from the results presented in Tables 2, 3 and 4, we experienced a very good performance of *SemanticRank* in ranking sentences for the text summarization task, with the weighted ranking variations producing always better results than the unweighted.

6 Conclusions and Future Work

In this paper we introduced *SemanticRank*, a new algorithm for ranking keywords and text segments using measures of semantic relatedness. The novelty of the algorithm is its semantic graph creation step, which is based on a measure of semantic relatedness that combines *WordNet* and *Wikipedia*. We evaluated *SemanticRank* using several alternatives for its ranking step, all based on weighted and unweighted variations of *PageRank* and *HITS*. Results in keyword extraction and text summarization experiments show that it performs favorably over state of the art related methods, and that the selected edges' weighting boosts its performance. In our future work we will examine the potentiality of more graph-based ranking methods, and it is on our next plans to embed *SemanticRank* on more linguistic tasks, such as sentiment analysis and opinion mining.

References

Agirre, Eneko and Aitor Soroa. 2009. Personalizing pagerank for word sense disambiguation. In *Proc. of the 12th Conference of the European Chapter of the ACL*, pages 33–41.

Aizawa, Akiko N. 2003. An information-theoretic perspective of tf-idf measures. *Information Processing Management*, 39(1):45–65.

Amini, Massih and Nicolas Usunier. 2007. A contextual query expansion approach by term clustering for robust text summarization. In *Proc. of the DUC 2007 Conference*.

Baldwin, Breck and Thomas S. Morton. 1998. Dynamic coreference-based summarization. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*.

Boguarev, Branimir and Christopher Kennedy. 1997. Salience-based content characterization of text documents. In *Proc. of ACL/EACL Workshop on Intelligent Scalable Text Summarization*.

Budanitsky, Alexander and Graeme Hirst. 2006. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13–47.

Gabrilovich, Evgeniy and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proc. of the 20th International Joint Conference on Artificial Intelligence*, pages 1606–1611.

Halliday, Michael A.K. and Ruqaiya Hasan. 1976. *Cohesion in English*. Longman.

Haveliwala, Taher H. 2002. Topic-sensitive pagerank. In *Proc. of the World Wide Web Conference*, pages 517–526.

Hickl, Andrew, Kirk Roberts, and Finley Lacatusu. 2007. Lcc's gistexter at duc 2007: Machine reading for update summarization. In *Proc. of the DUC 2007 Conference*.

Huang, Chong, Yong Hong Tian, Zhi Zhou, Charles X. Ling, and Tiejun Huang. 2006. Keyphrase extraction using semantic networks structure analysis. In *Proc. of the 6th International Conference on Data Mining*, pages 275–284.

Hulth, Anette. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, pages 216–223.

Lin, Chin-Yew and Eduard H. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proc. of the Human Language Technology Conference and the North American Chapter of the ACL Conference*.

Litvak, Marina and Mark Last. 2008. Graph-based keyword extraction for single-document summarization. In *Proc. of the ACL Workshop on Multi-source Multilingual Information Extraction and Summarization*, pages 17–24.

Mani, Interjeet and Eric Bloedorn. 1998. Machine learning of generic and user-focused summarization. In *Proc. of the 15th National Conference on A.I. and 10th Innovative Applications of A.I. Conference*, pages 821–826.

Mihalcea, Rada and Paul Tarau. 2004. TextRank: Bringing order into texts. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, pages 404–411.

Mihalcea, Rada. 2004. Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *Proc. of the 42nd Annual Meeting of the ACL*.

Milne, David N. and Ian H. Witten. 2008. An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In *Proc. of the 1st AAAI Workshop on Wikipedia and Artificial Intelligence*.

Salton, Gerard, Amit Singhal, Mandar Mitra, and Chris Buckley. 1997. Automatic text structuring and summarization. *Information Processing Management*, 33(2):193–207.

Steinberger, Josef and Karel Jezek. 2009. Text summarization: An old challenge and new approaches. In *Foundations of Computational Intelligence (6)*, pages 127–149.

Tsatsaronis, George, Iraklis Varlamis, and Michalis Vazirgiannis. 2010. Text relatedness based on a word thesaurus. *Journal of Artificial Intelligence Research*, 37:1–39.

Wang, Jinghua, Jianyi Liu, and Cong Wang. 2007. Keyword extraction based on pagerank. In *Proc. of the 11th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, pages 857–864.

White, Scott and Padhraic Smyth. 2003. Algorithms for estimating relative importance in networks. In *Proc. of the 9th International Conference on Knowledge Discovery and Data Mining*, pages 266–275.

Yeh, Jen-Yuan, Hao-Ren Ke, and Wei-Pang Yang. 2008. iSpreadRank: Ranking sentences for extraction-based summarization using feature weight propagation in the sentence similarity network. *Expert Syst. Appl.*, 35(3):1451–1462.

Chinese CCGbank: extracting CCG derivations from the Penn Chinese Treebank

Daniel Tse and **James R. Curran**
School of Information Technologies
University of Sydney
{dtse6695,james}@it.usyd.edu.au

Abstract

Automated conversion has allowed the development of wide-coverage corpora for a variety of grammar formalisms without the expense of manual annotation. Analysing new languages also tests formalisms, exposing their strengths and weaknesses.

We present Chinese CCGbank, a 760,000 word corpus annotated with Combinatory Categorial Grammar (CCG) derivations, induced automatically from the Penn Chinese Treebank (PCTB). We design parsimonious CCG analyses for a range of Chinese syntactic constructions, and transform the PCTB trees to produce them. Our process yields a corpus of 27,759 derivations, covering 98.1% of the PCTB.

1 Introduction

An annotated corpus is typically used to develop statistical parsers for a given formalism and language. An alternative to the enormous cost of hand-annotating a corpus for a specific formalism is to convert from an existing corpus.

The Penn Treebank (PTB; Marcus et al., 1994) has been converted to HPSG (Miyao et al., 2004), LFG (Cahill et al., 2002), LTAG (Xia, 1999), and CCG (Hockenmaier, 2003). Dependency corpora, e.g. the German Tiger corpus, have also been converted (Hockenmaier, 2006). The Penn Chinese Treebank (PCTB; Xue et al., 2005) provides analyses for 770,000 words of Chinese. Existing PCTB conversions have targeted TAG (Chen et al., 2005) and LFG (Burke and Lam, 2004; Guo et al., 2007).

We present Chinese CCGbank, a Chinese corpus of CCG derivations automatically induced from the PCTB. Combinatory Categorial Grammar (CCG; Steedman, 2000) is a lexicalised grammar formalism offering a unified account of local and non-local dependencies. We harness the facilities of CCG to provide analyses of Chinese syntax including topicalisation, pro-drop, zero copula, extraction, and the 把 *ba-* and 被 *bei*-constructions.

Pushing the boundaries of formalisms by subjecting them to unfamiliar syntax also tests their universality claims. The freer word order of Turkish (Hoffman, 1996) and the complex morphology of Korean (Cha et al., 2002) led to the development of extensions to the CCG formalism.

We present our analysis of Chinese syntax under CCG, and provide an algorithm, modelled after Hockenmaier and Steedman (2007), to incrementally transform PCTB trees into CCG derivations. The algorithm assigns CCG categories which directly encode head and subcategorisation information. Instances of Chinese syntax demanding special analysis, such as extraction, pro-drop or topicalisation, are pin-pointed and given elegant analyses which exploit the expressivity of CCG.

Our conversion yields CCG analyses for 27,759 PCTB trees (98.1%). Coverage on lexical items, evaluated by 10-fold cross-validation, is 94.46% (by token) and 73.38% (by type).

We present the first CCG analysis of Chinese syntax and obtain a wide-coverage CCG corpus of Chinese. Highly efficient statistical parsing using a CCGbank has recently been demonstrated for English (Clark and Curran, 2007). Our Chinese CCGbank will enable the development of similarly efficient wide-coverage CCG parsers for Chinese.

2 Combinatory Categorial Grammar

CCG (Steedman, 2000) is a lexicalised grammar formalism, with a transparent syntax-semantics interface, a flexible view of constituency enabling concise accounts of various phenomena, and a consistent account of local/non-local dependencies.

It consists of *categories*, which encode the type and number of arguments taken by lexical items, and *combinators*, which govern the possible interactions between categories.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1083–1091,
Beijing, August 2010

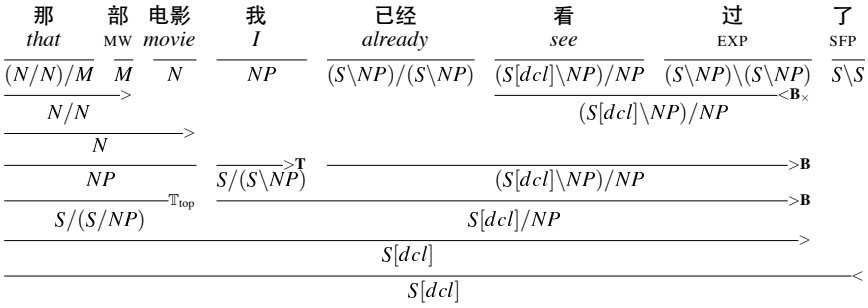


Figure 1: Chinese CCG derivation: "That movie, I've already seen."

A CCG grammar defines *atomic categories*, e.g. *NP* and *S*, which may be recursively constructed into *complex categories*, e.g. N/N and $S\backslash NP$.[1] Figure 1 shows how combinators govern the interaction of categories for lexical items, while slashes specify argument directionality.

The combinators allow us to reduce lexical ambiguity, by preserving a word's canonical category even when displaced from its canonical position. This facility is a strength of CCG, but elevates its generative power to mild context-sensitivity.

Some combinators may be disabled in a given language – the *multi-modal* CCG (Baldridge, 2002) allows these distinctions to be lexically specified.

Introducing non-CCG rules decrease categorial ambiguity at the expense of deviating from the formalism. Hockenmaier and Steedman (2002) show that these greatly improve lexical coverage. Their analysis of English employs non-CCG rules to coerce a verb phrase headed by a participle (category $S[ng]\backslash NP$) to a post-nominal modifier:

$$S[ng]\backslash NP \longrightarrow NP\backslash NP \quad (1)$$

This frees verbs from having to possess a distinct category in each position, thus trading off lexical ambiguity for derivational ambiguity. Honnibal and Curran (2009) extended CCG with *hat categories*, enabling the lexical specification of these unary type-change rules.

Hockenmaier and Steedman (2002, 2007) developed CCGbank, the first wide-coverage English CCG corpus, by converting 1.2 million words from the Wall Street Journal section of the PTB. CCGbank has made possible the development of wide-coverage statistical parsers for CCG in English, notably C&C (Clark and Curran, 2007).

[1] *Abbreviations in this paper:* The directionless slash | stands for one of $\{/, \backslash\}$. We also use the verbal category abbreviations $VP \equiv S\backslash NP$ and $TV \equiv (S\backslash NP)/NP$.

3 Penn Chinese Treebank

Xue et al. (2005) developed the Penn Chinese Treebank (PCTB), the first syntactically annotated corpus for Chinese. The corpus includes newswire text, magazine articles, and transcribed speech.[2]

Xue et al. establishes several principles for a more disciplined and consistent style of annotation compared to the original PTB. These principles include *complement/adjunct marking:* allowing the recovery of predicate-argument structure; *limited semantic role marking:* the annotation of modifier phrases with semantic roles; *covert argument marking:* the retention of traces of arguments deleted through pro-drop; and *NP internal structure:* bracketing of NP structure where the intended interpretation is clear.

The *one relation per bracketing* principle unambiguously encodes a grammatical relation (chiefly, predication, adjunction, or complementation) through the configuration of a node and its children. Xue et al. developed this principle to assist conversions from the PTB, e.g. Hockenmaier (2003), in resolving argument/adjunct distinctions.

PCTB derivations are pre-segmented, pre-tokenised, and POS tagged. Owing to the dearth of morphology in Chinese, the concept of *part of speech* is more fluid than that of English – the word 比较 *bijiao* 'compare' might be glossed as a verb, adjective, adverb, or noun depending on its context. Noun/verb mis-taggings are a frequent error case for PCFG parsing on PCTB data, compounded in Chinese by the lack of function words and morphology (Levy and Manning, 2003). This ambiguity is better handled by the adaptive multitagging approach used by Clark and Curran (2007) for CCG supertagging, in which each lexical item is tagged with a set of CCG categories.

We present our CCG analysis of Chinese syntax below, followed by our conversion algorithm.

[2] We use the Penn Chinese Treebank 6.0 (LDC2007T36).

4 The syntax of Chinese

4.1 Basic clause structure

Chinese is typologically SVO, with some OV elements (relative clauses, adjunct PPs and noun modifiers precede their heads). Numbers and determiners may not modify nouns directly; a *measure word* must intervene.

The category structure of the grammar may be inferred directly from headedness information. Heads subcategorise for the type, number and directionality of their arguments, while adjuncts receive modifier categories of the form $X \mid X$.

(2)

我	在		超市		买
I	at		supermarket		buy
NP	$(VP/VP)/NP$		*NP*		VP/NP
了	一		盒	鸡蛋	
PERF	one		box:MW	eggs	
$VP\backslash VP$	$(N/N)/M$		*M*	*N*	

I bought a box of eggs at the supermarket.

4.2 Topicalisation

In topic-prominent languages, the *topic* refers to information which the speaker assumes is known by the listener. In Mandarin, topicalisation manifests as left-dislocation of the topic phrase (Li and Thompson, 1989). We distinguish *gap* and *non-gap* topicalisation depending on whether the topic is co-referent with a gap in the sentence.[3]

For gapped topicalisation (cf. Figure 1), we adopt the Steedman (1987) topicalisation analysis:

$$T \rightarrow S/(S/T) \text{ for parametrically licensed } T \quad (3)$$

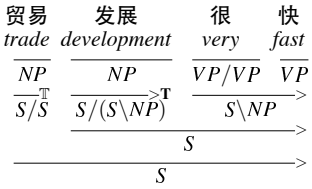
For non-gap topicalisation (Example 5), we use a variation of the analysis described in Hockenmaier and Steedman (2005), which treats the topicalised constituent as a sentential modifier. Under this analysis, the determiner in a topicalised *NP* receives $(S/S)/N$ instead of its canonical category NP/N. Instead, we propose a unary rule:

$$T \rightarrow S/S \text{ for topicalisation candidate } T \quad (4)$$

This delays the coercion to sentential modifier type (i.e. $NP \rightarrow S/S$) until after the NP has been consolidated, allowing the words under the topicalised NP to preserve their canonical categories.

[3]Non-gap topicalisation is also known as the *double subject construction* (Li and Thompson, 1989).

(5) (As for) trade, it has developed rapidly.



Topicalisation is far less marked in Chinese than in English, and the structure of topicalised constituents is potentially quite complex. The additional categorial ambiguity in Hockenmaier and Steedman (2005) compounds the data sparsity problem, leading us to prefer the unary rule.

4.3 Pro-drop

Since Chinese exhibits *radical pro-drop* (Neeleman and Szendrői, 2007), in which the viability of the pro-drop is not conditioned on the verb, the categorial ambiguity resulting from providing an additional argument-dropped category for every verb is prohibitive.

Rather than engendering sparsity on verbal categories, we prefer derivational ambiguity by choosing the unary rule analysis $S[dcl] \mid NP \rightarrow S[dcl]$ to capture Chinese pro-drop.

4.4 Zero copula

Although the Chinese copula 是 *shi* is obligatory when equating NPs, it may be omitted when equating an NP and a QP or PP (Tiee and Lance, 1986).[4]

(6)

她	今年	十八	岁
NP	*VP*/*VP*	$(S\backslash NP)/M$	*M*
3SG	this-year	18	years-old

She is 18 this year.

A solution involving a binary rule $NP\ QP \rightarrow S[dcl]$ is not properly headed, and thus violates the Principle of Lexical Head Government (Steedman, 2000). Conversely, a solution where, for example, 十八 '18' would have to receive the category $(S[dcl]\backslash NP)/M$ instead of its canonical category QP/M would lead to both data sparsity and over-generation, with VP modifiers becoming able to modify the QP directly. Tentatively, we ignore the data sparsity consequences, and have 十八 '18' receive the category $(S[dcl]\backslash NP)/M$ in this context.

[4]The copula is ungrammatical in predication on an adjectival verb, such as 高兴 'happy'. However, we analyse such words as verbs proper, with category $S[dcl]\backslash NP$.

4.5 把 *ba*- and 被 *bei*-constructions

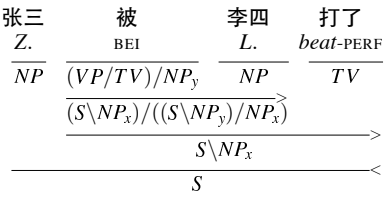
被 *bei* and 把 *ba* introduce a family of passive-like constructions in Chinese. Although superficially similar, the resulting constructions exhibit distinct syntax, as our CCG analysis reflects and clarifies.

In the 被 *bei*-construction, the patient argument of a verb moves to subject position, while the agent either becomes the complement of a particle 被 *bei* (the *long passive*), or disappears (the *short passive*; Yip and Rimmington, 1997). Although the two constructions are superficially similar (apparently differing only by the deletion of the agent NP), they behave differently in more complex contexts (Huang et al., 2008).

The long passive occurs with or without an object gap (deleted by identity with the subject of the matrix verb). We analyse this construction by assigning 被 *bei* a category which permutes the surface positions of the agent and patient. Coindexation of heads allows us to express long-distance dependencies.

Bei receives $((S\backslash NP_y)/((S\backslash NP_x)/NP_y))/NP_x$ in the gapped case (cf. Example 7) and $((S\backslash NP)/(S\backslash NP_x))/NP_x$ in the non-gapped case.

(7) Zhangsan was beaten by Lisi.



Short passives also occur with or without an object gap, receiving $(S\backslash NP_x)/((S\backslash NP)/NP_x)$ in the gapped case and $(S\backslash NP)\backslash(S\backslash NP)$ in the non-gapped case. Our analysis agrees with Huang et al. (2008)'s observation that short-*bei* is isomorphic to English *tough*-movement: our short-*bei* category is the same as Hockenmaier and Steedman (2005)'s category for English *tough*-adjectives.

In the 把 *ba* construction, a direct object becomes the complement of the morpheme 把 *ba*, and gains semantics related to "being affected, dealt with, or disposed of" (Huang et al., 2008). As for 被 *bei*, we distinguish two variants depending on whether the object is deleted under coreference with the complement of 把 *ba*.

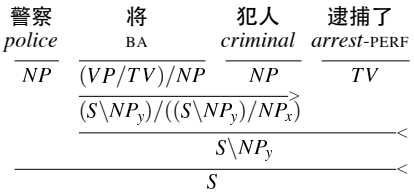
Ba receives $((S\backslash NP_y)/((S\backslash NP_y)/NP_x))/NP_x$ in the gapped case (cf. Example 8), and $((S\backslash NP_y)/(S\backslash NP_y))/NP$ in the non-gapped case.

As Levy and Manning (2003) suggest, we reshape the PCTB analysis of the *ba*-construction so that *ba* subcategorises for its NP and VP, rather than subcategorising for an IP sibling, which allows the NP to undergo extraction.

Tag	*Headedness*	*Example*
VSB	head-final	规划 建设 'plan [then] build'
VRD	right-adjunction	煮 熟 'cook done'
VCP	head-initial	确认 为 'confirm as'
VCD	appositive	投资 设厂 'invest [&] build-factory'
VNV	special	去 不 去 'go [or] not go'
VPT	special	离 得 开 'leave able away'

Table 1: Verb compounds in PCTB

(8) The criminals were arrested by the police.



4.6 Verbal compounding

Verbs resulting from compounding strategies are tagged and internally bracketed. Table 1 lists the types distinguished by the PCTB, and the headedness we assign to compounds of each type.

Modifier-head compounds (PCTB tag VSB) exhibit clear head-final semantics, with the first verb V_1 causally or temporally preceding V_2. Verb coordination compounds (VCD) project multiple heads, like ordinary lexical coordination.

In a resultative compound (VRD), the result or direction of V_1 is indicated by V_2, which we treat as a post-verbal modifier. The *V-not-V* construction (VNV) forms a yes/no question where $V_1 = V_2$. In the *V-bu/de-V* or potential verb construction (VPT), a disyllabic verb $V = V_1V_2$ receives the infix 得 *de* or 不 *bu* with the meaning *can/cannot* V. In both these cases, it is the infixed particle 得 *de* or 不 *bu* which collects its arguments on either side.

4.7 Extraction

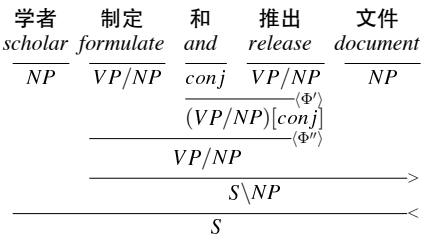
In the Chinese relative clause construction, the particle 的 *de* links a sentence with a subject or object gap with a NP to which that gap co-refers, in an analysis similar to the English construction described by Hockenmaier and Steedman (2005), mediated by the relative pronoun *that*.

As in the English object extraction case, forward type-raising on the subject argument, and forward composition into the verbal category allows us to obtain the correct object gap category S/NP.

4.8 Right node raising

Two coordinated verbs may share one or more contiguous arguments under right node raising. This analysis follows directly from the CCG definition of coordination, requiring no new lexical categories.

(9) Scholars have formulated and are releasing the documents.



4.9 Apposition

Apposition is the juxtaposition of two phrases referring to the same entity. Unlike noun modification, no clear modification relationship holds between the two phrases. The direct juxtaposition rules out Hockenmaier's (2003) analysis where a delimiting comma mediates the apposition. Chinese also allows full sentence/NP apposition:

(10) (用户 浪费 水)$_S$ 事件$_{NP}$
(users waste water)$_S$ incident$_{NP}$
incidents of users wasting water

This gives rise to the Chinese apposition binary rules $NP\ NP \rightarrow NP$ and $S[dcl]\ NP \rightarrow NP$.

5 The translation pipeline

5.1 Tagging

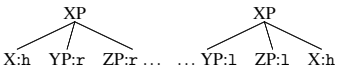
Each PCTB internal node structurally encodes a *configuration*, which lets us distinguish head-initial and head-final complementation from adjunction and predication (Xue et al., 2000).

The tagging mechanism annotates the PCTB tag of each internal node with a *marker*, which preserves this headedness information, even after the nodes are re-structured in the binarisation phase.

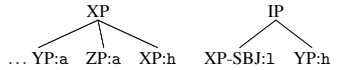
Hockenmaier's (2003) conversion algorithm uses the Magerman (1994) head-finding heuristics, a potential source of noise. Fortunately, the PCTB encodes gold standard headedness data.

The tagging algorithm is straightforward: if a node and its children unify with one of the schemata below, then the markers (e.g. `:l` or `:n`) are attached to its children. The markers `l` and `r` indicate complements *left*, or *right* of the *head* `h`; adjuncts are marked with `a`.

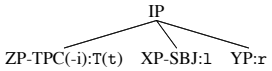
Head-initial, -final complementation



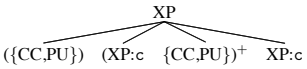
Adjunction, predication



Topicalisation (gap and non-gap)



Coordination



Others identify nodes with special syntax, such as topicalisation (`t`/`T`), apposition (`A`) or coordination (`c`), for special treatment in following phases.

NP internal structure

To speed annotation, NP internal structure is often left underspecified in PCTB (Xue et al., 2005), as in the Penn Treebank. As a result, 68% of non-trace `NP`s in PCTB have only a flat bracketing.

We assume that the internal structure of flat NPs is right-branching and head-final (Li and Thompson, 1989), following Hockenmaier and Steedman (2005), who assume this structure for English. A re-analysis of PCTB, like Vadas and Curran (2007) for the PTB, could restore this structure, and allow our conversion algorithm to yield the correct CCG analysis with no further modifications.

To obtain this default analysis, each node under NP internal structure receives the marker `n`, except the the final node, the head, which receives `N`.

5.2 Binarisation

CCG combinators take at most two categories, inducing binary derivation trees. As such, PCTB trees must be re-shaped to accommodate a CCG analysis.

Our markers control the shape of the binarised structure: head-initial complementation yields a left-branching tree, while head-final complementation, adjunction, predication, coordination, and NP internal structure all yield right-branching trees. Following Hockenmaier (2003), sentence-final punctuation is attached high.

Although the distinction between word-level tags (such as `NN`, `VA`) and phrasal tags (such as `NP`, `VP`, `LCP`) enables the configurational encoding of grammatical relations, it leaves a large number of

VP	←	VV,VE,VA,VRD	ADJP	←	JJ
ADVP	←	AD, CS	CLP	←	M
LCP	←	LC	DP	←	DT, OD
LST	←	OD	INTJ	←	IJ
FLR	←	*any node*	PP	←	P

Figure 2: Pruned unary projections

LCP	Localiser phrase	PP	Prepositional phrase
M	Measure word	QP	Quantifier phrase
N	Bare noun	S	Sentence
NP	Noun phrase	conj	Conjunction word

Table 2: Chinese CCGbank atomic category set

unary projections. While an intransitive verb (e.g. 睡觉 'sleep') would carry the verbal PCTB tag VV, and a transitive verb combined with its object (e.g. 吃了晚饭 'ate dinner') is annotated as VP, under CCG's freer concept of constituency, both receive the category $S\backslash NP$.

Pruning the unary projections in Fig. 2 prevents spurious category labellings in the next phase.

5.3 Labelling

We label each node of the binarised tree with CCG categories, respecting the headedness information encoded in the markers.

Atomic categories

The chosen mapping from PCTB tags to categories defines the *atomic category set* for the grammar. The richer representation in CCG categories permits some constituents to be expressed using a smaller set of atoms (e.g. an adjective is simply a noun modifier – N/N). Despite their critical importance in controlling the degree of under-/over-generation in the corpus, little guidance exists as to the selection of atomic categories in a CCG grammar. We observed the following principles:

Modifier proliferation: when two classes of words can be modified by the same class of modifiers, they should receive a single category;

Over-generation: the atom set should not overgeneralise to accept ungrammatical examples;

Efficiency: the representation may be motivated by the needs of applications such as parsers.

Table 2 shows the eight atomic categories chosen for our corpus. Two of these categories: *LCP* (localisers) and *M* (measure words) have variously been argued to be special sub-classes of nouns (Huang et al., 2008). However, based on our overgeneration criterion, we decided to represent these as atomic categories.

We adopt the bare/non-bare noun distinction from Hockenmaier and Steedman (2007) on parsing efficiency grounds. Although they roughly correspond to English *PP*s, the distributional differences between *PP*s, *LCP*s and *QP*s justify their inclusion as atoms in Chinese. Future work in training a wide-coverage parser on Chinese CCGbank will evaluate the impact of these choices.

Labelling algorithm

We developed a recursive algorithm which applies one of several labelling functions based on the markers on a node and its children.

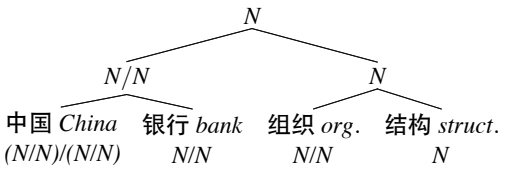
The algorithm proceeds top-down and assigns a CCG category to every node. The markers on a node's children are matched against the schema of Table 3, applying the categories of the matching schema to the children. The algorithm is then called recursively on each child. If the algorithm is called on an unlabelled node, the mapping from PCTB tags is used to assign a CCG category.

Predication	$C \to L\ C\backslash L$	Left absorption	$C \to p\ C$
Adjunction	$C \to C/C{:}\mathtt{a}\ C$	Right absorption	$C \to C\ p$
Right adjunction	$C \to C\ C\backslash C{:}\mathtt{a}$	Coordination	$C \to C{:}\mathtt{c}\ C[conj]$
Head-initial	$C \to C/R{:}\mathtt{h}\ R$	Partial coordination	$C[conj] \to conj\ C{:}\mathtt{c}$
Head-final	$C \to L\ C\backslash L{:}\mathtt{h}$	Apposition	$NP \to XP{:}\mathtt{A}\ NP$

Table 3: Category labelling schemata

Left- and right-absorption are non-CCG rules which functionally ignore punctuation, assuming that they project no dependencies and combine to yield the same category as their non-punctuation sibling (Hockenmaier and Steedman, 2007). In the schema, p represents a PCTB punctuation POS tag.

NPs receive a head-final bracketing (by our right-branching assumption), respecting NP internal structure where provided by PCTB:



6 Post-processing

A number of cases remain which are either not covered by the general translation algorithm, or otherwise could be improved in a post-processing step. The primary disharmony at this stage is the presence of *traces*, the empty categories which the PCTB annotation style uses to mark the canonical position of extraposed or deleted constituents. 19,781 PCTB derivations (69.9%) contain a trace. Since CCG aims to provide a transparent interface between surface string syntax and semantics, traces are expressly disallowed (Steedman, 2000). Hence, we eliminate traces from the annotation, by devising alternate analyses in terms of categories and combinatory rules.

Subject/object extraction

8966 PCTB derivations (31.7%) contain a subject extraction, while 3237 (11.4%) contain an object extraction. Figure 3 shows the canonical representation of subject extraction in the PCTB annotation style. The PCTB annotation follows the X' analysis of the relative clause construction as described by Wu (2004), which we transform into an equivalent, trace-free CCG analysis.

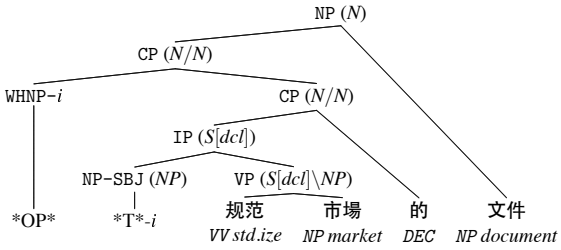


Figure 3: '*the document which standardises the market*'

First, the *Spec* trace, WHNP-i, coindexed with the extracted argument(s), is deleted. Next, the extracted argument(s) with matching indices are deleted, and category structure is adjusted to generate the correct gap category.

Modifier categories

Under our analysis, aspect particles such as 了 *le* (perfective) and 过 *guo* (experiential) are verbal post-modifiers, corresponding to *right adjunction* in Table 3. Accordingly, an aspect particle following a transitive verb VP/NP will receive the modifier category $(VP/NP)\backslash(VP/NP)$. Under this analysis, every verbal category gives rise to one possible modifier category for each aspect particle, leading to detrimental categorial ambiguity.

However, the generalised backward crossed composition combinator (Steedman, 2000) lets aspect particles retain their canonical category $(S\backslash NP)\backslash(S\backslash NP)$ regardless of the arity of the verb they modify.

Transformations

The PCTB annotation style posits traces to account for gapping, control/raising, argument sharing, pro-drop and topicalisation. To effect the parsimonious CCG analyses of Section 4, structural transformations on the original PCTB trees are necessary to accommodate the new analyses.

We developed a `tgrep`-like language which identifies instances of Chinese constructions, such as right node raising and pro-drop, whose PCTB annotation posits traces. The local trees are then reshaped to accommodate trace-free CCG analyses.

7 Evaluation

This section explores the coverage characteristics of Chinese CCGbank, in comparison with the English and German CCGbanks generated by Hockenmaier. Our analysis follows Hockenmaier (2006) in establishing *coverage* as the metric reflecting how well the target corpus has accounted for constructions in the source corpus.

7.1 Corpus coverage

The Chinese CCGbank conversion algorithm completes for 28,227 of the 28,295 (99.76%) PCTB trees. Annotation noise, and rare but legitimate syntax, such as ellipsis, account for the coverage lost in this phase. Following Hockenmaier and Steedman (2005), we adjust the PCTB annotation only for systematic tagging errors that lead to category mis-assignments, maintaining as far as possible the PCTB bracketing.

269 derivations (0.95%) contain unresolved traces, resulting from annotation noise and rare constructions (such as ellipsis) not currently handled by our translation algorithm. In 468 (1.66%) derivations, residues of PCTB tags not eliminated by the translation algorithm generate malformed categories outside the allowed set (Table 2). Excluding these cases, our conversion algorithm results in a corpus of 27,759 (98.1%) valid derivations.

7.2 Category set

The Chinese CCGbank category set is compared against existing CCG corpora derived from similar automatic corpus conversions, to determine how

well we have generalised over syntactic phenomena in the source corpus.

A total of 1197 categories appear in the final corpus, of which 329 occur at least ten times, and 478 are attested only once. By comparison, English CCGbank, contains 1286 categories, 425 of which occur at least ten times, and 440 only once, while German CCGbank has a category inventory of 2506 categories, with 1018 attested only once.[5]

7.3 Lexicon coverage

Lexical item coverage establishes the extent to which data sparsity due to unseen words is problematic in the source corpus, and hence in any corpus derived from it. Hockenmaier and Steedman (2001) showed that formalisms with rich tagsets, such as CCG, are particularly sensitive to this sparsity – while a lexical item may be attested in the training data, it may lack the necessary category.

We divided the 27,759 valid derivations into ten contiguous sections, performing ten-fold cross-validation to determine the coverage of lexical items and CCG categories in the resulting corpus.

Average coverage on lexical items is 73.38%, while average coverage on categories is 88.13%. 94.46% of token types from the held-out set are found in the training set. These figures compare to 86.7% lexical coverage (by type) and 92% (by token) in German CCGbank (Hockenmaier, 2006). Although lexical coverage by token is comparable to the German corpus, we observe a marked difference in coverage by type.

To explain this, we examine the most frequent POS tags among the missing tokens. These are NN (common nouns; 16,552 tokens), NR (proper noun; 8458), VV (verb; 6879), CD (numeral; 1814) and JJ (adjective; 1257). The 100 most frequent missing tokens across the ten folds comprise 48 NR tokens, 46 NR, 3 NT (temporal nouns), 2 JJ (adjectives) and one VA (verbal adjective). Personal names are also not tokenised into surnames and forenames in the PCTB, increasing unseen NR tokens.

The missing VVs (verbs) include 1342 *four-character compounds*, fossilised idiomatic expressions which are considered atomic verbs in the PCTB annotation. Another source of verb sparsity stems from the PCTB analysis of verbal infixation. Given a polysyllabic verb (e.g. 离开 *leave-away* "leave"), we can add the adverbial infix 不 *not* to form a potential verb 离不开 *leave-not-away* "unable to leave". In the PCTB annotation, however, this results in lexical items for the two cleaved parts, even though 离 *leave* can no longer stand alone as a verb in modern Chinese. In this case, a morphologically decomposed representation which does not split the lexical item could mitigate against this sparsity. Alternatively, candidate verbs for this construction could have the first verb fragment subcategorise for the second.

[5] All German verbs having at least two categories to account for German verbal syntax contributes to the greater size of the category set (Hockenmaier, 2006).

8 Conclusion

We have developed the first analysis of Chinese with Combinatory Categorial Grammar, crafting novel CCG analyses for a range of constructions including topicalisation, pro-drop, zero copula, verb compounding, and the long-range dependencies resulting from the 把 *ba-* and 被 *bei*-constructions.

We have presented an elegant and economical account of Chinese syntax that exploits the power of CCG combinatory rules, supporting Steedman's claim to its language-independence.

We have designed a conversion algorithm to extract this analysis from an existing treebank, avoiding the massive cost of hand re-annotation, creating a corpus of 27,759 CCG derivations, covering 98.1% of the PCTB. The corpus will be publicly released, together with the converter, providing the tools to create CCGbanks in new languages.

At release, Chinese CCGbank will include gold-standard head co-indexation data, as required for the training and evaluation of head-driven dependency parsers. Co-indexation analyses, like those provided for the 把 *ba-* and 被 *bei*-constructions, will be extended to all categories.

Future refinements which could be brought to bear on Chinese CCGbank include the integration of PropBank data into CCGbank (Honnibal and Curran, 2007; Boxwell and White, 2008) using Chinese PropBank (Xue, 2008). The *hat categories* of Honnibal and Curran (2009) may better handle form/function discrepancies such as the Chinese zero copula construction, leading to cleaner, more general analyses.

We have presented a wide-coverage Chinese corpus which exploits the strengths of CCG to analyse a range of challenging Chinese constructions. We are now ready to develop rich NLP tools, including efficient, wide-coverage CCG parsers, to address the ever-increasing volumes of Chinese text now available.

Acknowledgements

James Curran was supported by Australian Research Council (ARC) Discovery grant DP1097291 and the Capital Markets Cooperative Research Centre.

References

Jason Baldridge. 2002. *Lexically Specified Derivational Control in Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

Stephen Boxwell and Michael White. 2008. Projecting Propbank roles onto the CCGbank. *Proceedings of LREC 2008*.

Michael Burke and Olivia Lam. 2004. Treebank-based acquisition of a Chinese lexical-functional grammar. In *Proceedings of the 18th Pacific Asia Conference on Language, Information and Computation*, pages 161–172.

Aoife Cahill, Mairead McCarthy, Josef van Genabith, and Andy Way. 2002. Automatic annotation of the Penn Treebank with LFG F-structure information. In *LREC 2002 Workshop on Linguistic Knowledge Acquisition and Representation-Bootstrapping Annotated Language Data*, pages 8–15.

Jeongwon Cha, Geunbae Lee, and Jonghyeok Lee. 2002. Korean combinatory categorial grammar and statistical parsing. *Computers and the Humanities*, 36(4):431–453.

John Chen, Srinivas Bangalore, and K. Vijay-Shanker. 2005. Automated extraction of Tree-Adjoining Grammars from treebanks. *Natural Language Engineering*, 12(03):251–299.

Stephen Clark and James R. Curran. 2007. Wide-Coverage Efficient Statistical Parsing with CCG and Log-Linear Models. In *Computational Linguistics*, volume 33, pages 493–552.

Yuqing Guo, Josef van Genabith, and Haifeng Wang. 2007. Treebank-based acquisition of LFG resources for Chinese. In *Proceedings of LFG07 Conference*, pages 214–232.

Julia Hockenmaier. 2003. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

Julia Hockenmaier. 2006. Creating a CCGbank and a wide-coverage CCG lexicon for German. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 505–512. Morristown, NJ, USA.

Julia Hockenmaier and Mark Steedman. 2001. Generative models for statistical parsing with combinatory categorial grammar. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 335–342. Association for Computational Linguistics, Morristown, NJ, USA.

Julia Hockenmaier and Mark Steedman. 2002. Acquiring compact lexicalized grammars from a cleaner treebank. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, pages 1974–1981.

Julia Hockenmaier and Mark Steedman. 2005. CCGbank: Users' manual. Technical report, MS-CIS-05-09, Computer and Information Science, University of Pennsylvania.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Beryl Hoffman. 1996. *The computational analysis of the syntax and interpretation of free word order in Turkish*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

Matthew Honnibal and James R. Curran. 2007. Improving the complement/adjunct distinction in CCGbank. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics (PACLING-07)*, pages 210–217.

Matthew Honnibal and James R. Curran. 2009. Fully Lexicalising CCGbank with Hat Categories. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1212–1221.

C.-T. James Huang, Y.-H. Audrey Li, and Yafei Li. 2008. *The syntax of Chinese*. Cambridge University Press.

Roger Levy and Christopher Manning. 2003. Is it harder to parse Chinese, or the Chinese Treebank? In *Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 439–446. Morristown, NJ, USA.

Charles N. Li and Sandra A. Thompson. 1989. *Mandarin Chinese: A functional reference grammar*. University of California Press.

David M. Magerman. 1994. *Natural language parsing as statistical pattern recognition*. Ph.D. thesis, Stanford University.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1994. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Yusuke Miyao, Takashi Ninomiya, and Jun'ichi Tsujii. 2004. Corpus-Oriented Grammar Development for Acquiring a Head-Driven Phrase Structure Grammar from the Penn Treebank. pages 684–693.

Ad Neeleman and Kriszta Szendrői. 2007. Radical pro drop and the morphology of pronouns. *Linguistic Inquiry*, 38(4):671–714.

Mark Steedman. 1987. Combinatory grammars and parasitic gaps. *Natural Language & Linguistic Theory*, 5(3):403–439.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press. Cambridge, MA, USA.

Henry H.Y. Tiee and Donald M. Lance. 1986. *A reference grammar of Chinese sentences with exercises*. University of Arizona Press.

David Vadas and James R. Curran. 2007. Adding noun phrase structure to the Penn Treebank. In *Association for Computational Linguistics*, volume 45, page 240.

Xiu-Zhi Zoe Wu. 2004. *Grammaticalization and language change in Chinese: A formal view*. Routledge.

Fei Xia. 1999. Extracting tree adjoining grammars from bracketed corpora. In *Proceedings of Natural Language Processing Pacific Rim Symposium '99*, pages 398–403.

Nianwen Xue. 2008. Labeling chinese predicates with semantic roles. *Computational Linguistics*, 34(2):225–255.

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The Penn Chinese TreeBank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(02):207–238.

Nianwen Xue, Fei Xia, Shizhe Huang, and Anthony Kroch. 2000. The Bracketing Guidelines for the Penn Chinese Treebank (3.0). *IRCS Report 00-08, University of Pennsylvania*.

Po Ching Yip and Don Rimmington. 1997. *Chinese: An essential grammar*. Routledge.

Dependency Forest for Statistical Machine Translation

Zhaopeng Tu † **Yang Liu** † **Young-Sook Hwang** ‡ **Qun Liu** † **Shouxun Lin** †

†Key Lab. of Intelligent Info. Processing
Institute of Computing Technology
Chinese Academy of Sciences
{tuzhaopeng,yliu,liuqun,sxlin}@ict.ac.cn

‡HILab Convergence Technology Center
C&I Business
SKTelecom
yshwang@sktelecom.com

Abstract

We propose a structure called *dependency forest* for statistical machine translation. A dependency forest compactly represents multiple dependency trees. We develop new algorithms for extracting string-to-dependency rules and training dependency language models. Our forest-based string-to-dependency system obtains significant improvements ranging from 1.36 to 1.46 BLEU points over the tree-based baseline on the NIST 2004/2005/2006 Chinese-English test sets.

1 Introduction

Dependency grammars have become increasingly popular in syntax-based statistical machine translation (SMT). One important advantage of dependency grammars is that they directly capture the dependencies between words, which are key to resolving most parsing ambiguities. As a result, incorporating dependency trees proves to be effective in improving statistical machine translation (Quirk et al., 2005; Ding and Palmer, 2005; Shen et al., 2008).

However, most dependency-based translation systems suffer from a major drawback: they only use 1-best dependency trees for rule extraction, dependency language model training, and decoding, which potentially introduces translation mistakes due to the propagation of parsing errors (Quirk and Corston-Oliver, 2006). While the treelet system (Quirk et al., 2005) takes a dependency tree as input, the string-to-dependency system (Shen et al., 2008) decodes on a source-language string. However, as we will show, the string-to-dependency system still commits to using degenerate rules and dependency language models learned from noisy 1-best trees.

To alleviate this problem, an obvious solution is to offer more alternatives. Recent studies have shown that SMT systems can benefit from widening the annotation pipeline: using packed forests instead of 1-best trees (Mi and Huang, 2008), word lattices instead of 1-best segmentations (Dyer et al., 2008), and weighted alignment matrices instead of 1-best alignments (Liu et al., 2009).

Along the same direction, we propose a structure called *dependency forest*, which encodes exponentially many dependency trees compactly, for dependency-based translation systems. In this paper, we develop two new algorithms for extracting string-to-dependency rules and for training dependency language models, respectively. We show that using the rules and dependency language models learned from dependency forests leads to consistent and significant improvements over that of using 1-best trees on the NIST 2004/2005/2006 Chinese-English test sets.

2 Background

Figure 1 shows a dependency tree of an English sentence *he saw a boy with a telescope*. Arrows point from the child to the parent, which is often referred to as the head of the child. For example, in Figure 1, *saw* is the head of *he*. A dependency tree is more compact than its constituent counterpart because there is no need to build a large superstructure over a sentence.

Shen et al. (2008) propose a novel string-to-dependency translation model that features two important advantages. First, they define that a string-to-dependency rule must have a *well-formed* dependency structure on the target side, which makes efficient dynamic programming possible and manages to retain most useful non-constituent rules. A well-formed structure can be either *fixed* or *floating* . A fixed structure is a

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1092–1100,
Beijing, August 2010

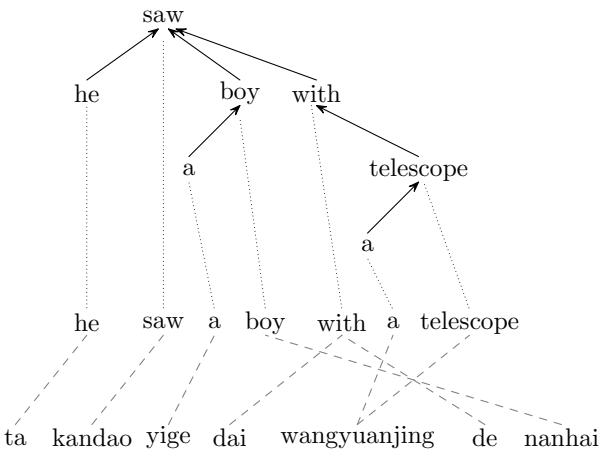


Figure 1: A training example for tree-based rule extraction.

dependency tree with all the children complete. Floating structures consist of sibling nodes of a common head, but the head itself is unspecified or floating. For example, Figure 2(a) and Figure 2(b) are two fixed structures while Figure 2(c) is a floating one.

Formally, for a given sentence $w_{1:l} = w_1 \dots w_l$, $d_1 \dots d_l$ represent the parent word IDs for each word. If w_i is a root, we define $d_i = 0$.

Definition 1. *A dependency structure $d_{i..j}$ is* **fixed on head** *h, where h $\notin [i, j]$, or* **fixed** *for short, if and only if it meets the following conditions*

- $d_h \notin [i, j]$
- $\forall k \in [i, j]$ and $k \neq h, d_k \in [i, j]$
- $\forall k \notin [i, j]$, $d_k = h$ or $d_k \notin [i, j]$

Definition 2. *A dependency structure $d_{i..j}$ is* **floating with children** *C, for a non-empty set C $\subseteq \{i, ..., j\}$, or* **floating** *for short, if and only if it meets the following conditions*

- $\exists h \notin [i, j], s.t. \forall k \in C, d_k = h$
- $\forall k \in [i, j]$ and $k \notin C, d_k \in [i, j]$
- $\forall k \notin [i, j]$, $d_k \notin [i, j]$

A dependency structure is **well-formed** if and only if it is either **fixed** or **floating**.

2.1 Tree-based Rule Extraction

Figure 1 shows a training example consisting of an English dependency tree, its Chinese translation,

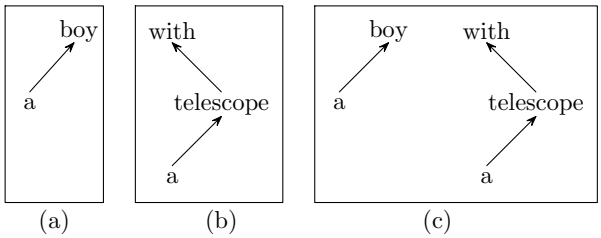


Figure 2: Well-formed dependency structures corresponding to Figure 1. (a) and (b) are fixed and (c) is floating.

and the word alignments between them. To facilitate identifying the correspondence between the English and Chinese words, we also gives the English sentence. Extracting string-to-dependency rules from aligned string-dependency pairs is similar to extracting SCFG (Chiang, 2007) except that the target side of a rule is a well-formed structure. For example, we can first extract a string-to-dependency rule that is consistent with the word alignment (Och and Ney, 2004):

with ((*a*) *telescope*) $\rightarrow$ *dai wangyuanjing de*

Then a smaller rule

(*a*) *telescope* $\rightarrow$ *wangyuanjing*

can be subtracted to obtain a rule with one non-terminal:

with (X_1) $\rightarrow$ *dai* X_1 *de*

where X is a non-terminal and the subscript indicates the correspondence between non-terminals on the source and target sides.

2.2 Tree-based Dependency Language Model

As dependency relations directly model the semantics structure of a sentence, Shen et al. (2008) introduce *dependency language model* to better account for the generation of target sentences. Compared with the conventional n-gram language models, dependency language model excels at capturing non-local dependencies between words (e.g., *saw* ... *with* in Figure 1). Given a dependency tree, its dependency language model probability is a product of three sub-models defined between headwords and their dependants. For example, the probability of the tree in Figure 1 can

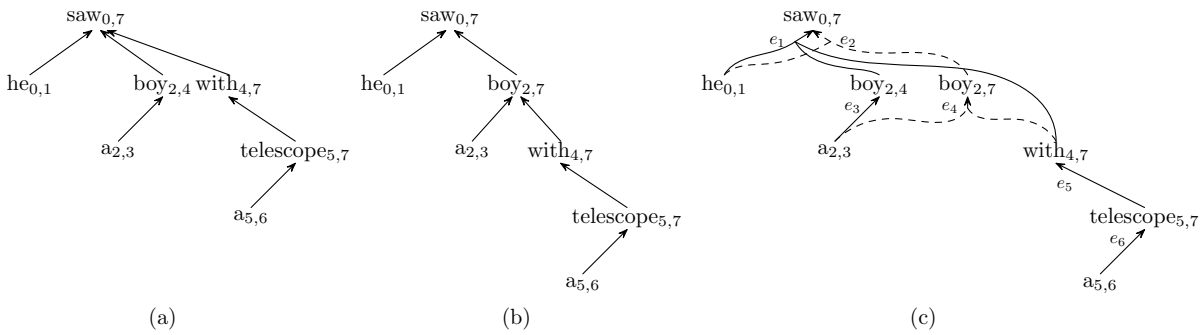


Figure 3: (a) the dependency tree in Figure 1, (b) another dependency tree for the same sentence, and (c) a dependency forest compactly represents the two trees.

be calculated as:

$$\begin{aligned} Prob \;=\; & P_T(saw) \\ & \times P_L(he|saw\text{-as-head}) \\ & \times P_R(boy|saw\text{-as-head}) \\ & \times P_R(with|boy, saw\text{-as-head}) \\ & \times P_L(a|boy\text{-as-head}) \\ & \times P_R(telescope|with\text{-as-head}) \\ & \times P_L(a|telescope\text{-as-head}) \end{aligned}$$

where $P_T(x)$ is the probability of word x being the root of a dependency tree. P_L and P_R are the generative probabilities of left and right sides respectively.

As the string-to-tree system relies on 1-best trees for parameter estimation, the quality of rule table and dependency language model might be affected by parsing errors and therefore ultimately results in translation mistakes.

3 Dependency Forest

We propose to encode multiple dependency trees in a compact representation called dependency forest, which offers an elegant solution to the problem of parsing error propagation.

Figures 3(a) and 3(b) show two dependency trees for the example English sentence in Figure 1. The prepositional phrase *with a telescope* could either depend on *saw* or *boy*. Figure 3(c) is a dependency forest compactly represents the two trees by sharing common nodes and edges.

Each **node** in a dependency forest is a word. To distinguish among nodes, we attach a **span** to each node. For example, in Figure 1, the span of the first *a* is $(2, 3)$ because it is the third word in the sentence. As the fourth word *boy* dominates the node $a_{2,3}$, it can be referred to as $boy_{2,4}$. Note that the position of *boy* itself is taken into consideration. Similarly, the word *boy* in Figure 3(b) can be represented as $boy_{2,7}$.

The nodes in a dependency forest are connected by **hyperedges**. While an edge in a dependency tree only points from a dependent to its head, a hyperedge groups all the dependants that have a common head. For example, in Figure 3(c), the hyperedge

$$e_1\colon \langle (he_{0,1}, boy_{2,4}, with_{4,7}), saw_{0,7} \rangle$$

denotes that $he_{0,1}$, $boy_{2,4}$, and $with_{4,7}$ are dependants (from left to right) of $saw_{0,7}$.

More formally, a *dependency forest* is a pair $\langle V, E \rangle$, where V is a set of nodes, and E is a set of hyperedges. For a given sentence $w_{1:l} = w_1 \dots w_l$, each node $v \in V$ is in the form of $w_{i,j}$, which denotes that w dominates the substring from positions i through j (i.e., $w_{i+1} \dots w_j$). Each hyperedge $e \in E$ is a pair $\langle tails(e), head(e) \rangle$, where $head(e) \in V$ is the head and $tails(e) \in V$ are its dependants.

A dependency forest has a structure of a *hypergraph* such as packed forest (Klein and Manning, 2001; Huang and Chiang, 2005). However, while each hyperedge in a packed forest naturally treats the corresponding PCFG rule probability as its weight, it is challenging to make dependency forest to be a weighted hypergraph because dependency parsers usually only output a score, which can be either positive or negative, for each edge in a dependency tree rather than a hyperedge in a

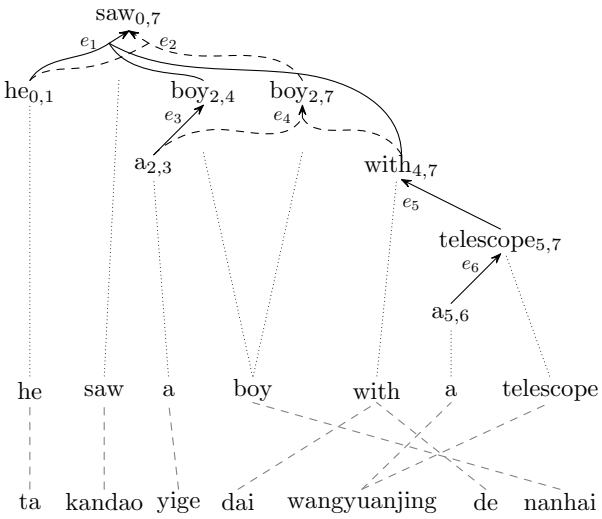


Figure 4: A training example for forest-based rule extraction.

dependency forest. For example, in Figure 3(a), the scores for the edges *he* $\rightarrow$ *saw*, *boy* $\rightarrow$ *saw*, and *with* $\rightarrow$ *saw* could be 13, 22, and -12, respectively.

To assign a probability to each hyperedge, we can first obtain a positive number for a hyperedge using the scores of the corresponding edges:[1]

$$c(e) = exp\left(\frac{\sum_{v \in tails(e)} s\big(v, head(e)\big)}{|tails(e)|}\right) \quad (1)$$

where $c(e)$ is the count of a hyperedge e, $head(e)$ is a head, $tails(e)$ is a set of dependants of the head, v is one dependant, and $s(v, head(e))$ is the score of an edge from v to $head(e)$. For example, the count of the hyperedge e_1 in Figure 3(c) is

$$c(e_1) = exp\left(\frac{13 + 22 - 12}{3}\right) \quad (2)$$

Then, the probability of a hyperedge can be obtained by normalizing the count among all hyperedges with the same head collected from a training corpus:

$$p(e) = \frac{c(e)}{\sum_{e': head(e') = head(e)} c(e')} \quad (3)$$

Therefore, we obtain a weighted dependency forest in which each hyperedge has a probability.

[1]It is difficult to assign a probability to each hyperedge. The current method is arbitrary, and we will improve it in the future.

Algorithm 1 Forest-based Initial Phrase Extraction

Input: a source sentence ψ, a forest F, an alignment a, and k
Output: minimal initial phrase set $\mathcal{R}$

for each node $v \in V$ in a bottom-up order **do**
  for each hyperedge $e \in E$ and $head(e) = v$ **do**
    $W \leftarrow \emptyset$
    fixs $\leftarrow$ *EnumFixed*$(v, modifiers(e))$
    floatings $\leftarrow$ *EnumFloating*$(modifiers(e))$
    add structures *fixs*, *floatings* to W
    for each $\omega \in W$ **do**
      if ω is consistent with a **then**
        generate a rule r
        $\mathcal{R}$.append(r)
  keep k-best dependency structures for v

4 Forest-based Rule Extraction

In tree-based rule extraction, one just needs to first enumerate all bilingual phrases that are consistent with word alignment and then check whether the dependency structures over the target phrases are well-formed. However, this algorithm fails to work in the forest scenario because there are usually exponentially many well-formed structures over a target phrase.

The GHKM algorithm (Galley et al., 2004), which is originally developed for extracting tree-to-string rules from 1-best trees, has been successfully extended to packed forests recently (Mi and Huang, 2008). The algorithm distinguishes between minimal and composed rules. Although there are exponentially many composed rules, the number of minimal rules extracted from each node is rather limited (e.g., one or zero). Therefore, one can obtain promising composed rules by combining minimal rules.

Unfortunately, the GHKM algorithm cannot be applied to extracting string-to-dependency rules from dependency forests. This is because the GHKM algorithm requires a complete subtree to exist in a rule while neither fixed nor floating dependency structures ensure that all dependants of a head are included. For example, the floating structure shown in Figure 2(c) actually contains two trees.

Alternatively, our algorithm searches for well-formed structures for each node in a bottom-up style. Algorithm 1 shows the algorithm for extracting initial phrases, that is, rules without non-

terminals from dependency forests. The algorithm maintains k-best well-formed structures for each node (line 11). The well-formed structures of a head can be constructed from those of its dependants. For example, in Figure 4, as the fixed structure rooted at *telescope*$_{5,7}$ is

(*a*) *telescope*

we can obtain a fixed structure rooted for the node *with*$_{4,7}$ by attaching the fixed structure of its dependant to the node (*EnumFixed* in line 4). Figure 2(b) shows the resulting fixed structure.

Similarly, the floating structure for the node *saw*$_{0,7}$ can be obtained by concatenating the fixed structures of its dependants *boy*$_{2,4}$ and *with*$_{4,7}$ (*EnumFloating* in line 5). Figure 2(c) shows the resulting fixed structure. The algorithm is similar to Wang et al. (2007), which binarize each constituent node to create some intermediate nodes that correspond to the floating structures.

Therefore, we can find k-best fixed and floating structures for a node in a dependency forest by manipulating the fixed structures of its dependants. Then we can extract string-to-dependency rules if the dependency structures are consistent with the word alignment.

How to judge a well-formed structure extracted from a node is better than others? We follow Mi and Huang (2008) to assign a **fractional count** to each well-formed structure. Given a tree fragment t, we use the inside-outside algorithm to compute its posterior probability:

$$\begin{aligned}\alpha\beta(t) &= \alpha(root(t)) \times \prod_{e \in t} p(e) \\ &\quad \times \prod_{v \in leaves(t)} \beta(v) \end{aligned} \quad (4)$$

where $root(t)$ is the root of the tree, e is an edge, $leaves(t)$ is a set of leaves of the tree, $\alpha(\cdot)$ is outside probability, and $\beta(\cdot)$ is inside probability.

For example, the subtree rooted at *boy*$_{2,7}$ in Figure 4 has the following posterior probability:

$$\begin{aligned}&\alpha(\textit{boy}_{2,7}) \times p(e_4) \times p(e_5) \\ &\times p(e_6) \times \beta(\textit{a}_{2,3}) \times \beta(\textit{a}_{5,6}) \end{aligned} \quad (5)$$

Now the fractional count of the subtree t is

$$c(t) = \frac{\alpha\beta(t)}{\alpha\beta(TOP)} \quad (6)$$

where TOP denotes the root node of the forest.

As a well-formed structure might be non-constituent, we approximate the fractional count by taking that of the minimal constituent tree fragment that contains the well-formed structure. Finally, the fractional counts of well-formed structures can be used to compute the relative frequencies of the rules having them on the target side (Mi and Huang, 2008):

$$\phi(r|lhs(r)) = \frac{c(r)}{\sum_{r':lhs(r')=lhs(r)} c(r')} \quad (7)$$

$$\phi(r|rhs(r)) = \frac{c(r)}{\sum_{r':rhs(r')=rhs(r)} c(r')} \quad (8)$$

Often, our approach extracts a large amount of rules from training corpus as we usually retain exponentially many well-formed structures over a target phrase. To maintain a reasonable rule table size, we discard any rule that has a fractional count lower that a threshold t.

5 Forest-based Dependency Language Model Training

Dependency language model plays an important role in string-to-dependency system. Shen et al. (2008) show that string-to-dependency system achieves 1.48 point improvement in BLEU along with dependency language model, while no improvement without it. However, the string-to-dependency system still commits to using dependency language model from noisy 1-best trees. We now turn to dependency forest for it encodes multiple dependency trees.

To train a dependency language model from a dependency forest, we need to collect all heads and their dependants. This can be easily done by enumerating all hyperedges. Similarly, we use the inside-outside algorithm to compute the posterior probability of each hyperedge e,

$$\begin{aligned}\alpha\beta(e) &= \alpha(head(e)) \times p(e) \\ &\quad \times \prod_{v \in tailes(e)} \beta(v) \end{aligned} \quad (9)$$

For example, the posterior probability of the hyperedge e_2 in Figure 4 is calculated as

$$\begin{aligned}\alpha\beta(e_2) &= \alpha(\textit{saw}_{0,7}) \times p(e_2) \\ &\quad \times \beta(\textit{he}_{0,1}) \times \beta(\textit{boy}_{2,7}) \end{aligned} \quad (10)$$

Rule	DepLM	NIST 2004	NIST 2005	NIST 2006	time
tree	tree	33.97	30.21	30.73	19.6
tree	forest	34.42*	31.06*	31.37*	24.1
forest	tree	34.60*	31.16*	31.45*	21.7
forest	forest	**35.33****	**31.57****	**32.19****	28.5

Table 1: BLEU scores and average decoding time (second/sentence) on the Chinese-English test sets. The baseline system (row 2) used the rule table and dependency language model learned both from 1-best dependency trees. We use " *" and "**" to denote a result is better than baseline significantly at $p < 0.05$ and $p < 0.01$, respectively.

Then, we can obtain the fractional count of a hyperedge e,

$$c(e) = \frac{\alpha\beta(e)}{\alpha\beta(TOP)} \quad (11)$$

Each n-gram (e.g., "boy-as-head a") is assigned the same fractional count of the hyperedge it belongs to.

We also tried training dependency language model as in (Shen et al., 2008), which means all hyperedges were on equal footing without regarding probabilities. However, the performance is about 0.8 point lower in BLEU. One possbile reason is that hyperedges with probabilities could distinguish high quality structures better.

6 Experiments

6.1 Results on the Chinese-English Task

We used the FBIS corpus (6.9M Chinese words + 8.9M English words) as our bilingual training corpus. We ran GIZA++ (Och and Ney, 2000) to obtain word alignments. We trained a 4-gram language model on the Xinhua portion of GIGAWORD corpus using the SRI Language Modeling Toolkit (Stolcke, 2002) with modified Kneser-Ney smoothing (Kneser and Ney, 1995). We optimized feature weights using the minimum error rate training algorithm (Och and Ney, 2002) on the NIST 2002 test set. We evaluated the translation quality using case-insensitive BLEU metric (Papineni et al., 2002) on the NIST 2004/2005/2006 test sets.

To obtain dependency trees and forests, we parsed the English sentences of the FBIS corpus using a shift-reduce dependency parser that enables beam search (Huang et al., 2009). We only retained the best well-formed structure for each node when extracting string-to-tree rules from dependency forests (i.e., $k = 1$). We trained two 3-gram depLMs (one from trees and another from forests) on English side of FBIS corpus plus 2M sentence pairs from other LDC corpus.

Rules	Size	New Rules
tree	7.2M	-
forest	7.6M	16.86%

Table 2: Statistics of rules. The last column shows the ratio of rules extracted from non 1-best parses being used in 1-best derivations.

After extracting rules and training depLMs, we ran our replication of string-to-dependency system (Shen et al., 2008) to translate the development and test sets.

Table 1 shows the BLEU scores on the test sets. The first column "Rule" indicates where the string-to-dependency rules are learned from: 1-best dependency trees or dependency forests. Similarly, the second column "DepLM" also distinguish between the two sources for training dependency language models. The baseline system used the rule table and dependency language model both learned from 1-best dependency trees. We find that adding the rule table and dependency language models obtained from dependency forests improves string-to-dependency translation consistently and significantly, ranging from +1.3 to +1.4 BLEU points. In addition, using the rule table and dependency language model trained from forest only increases decoding time insignificantly.

How many rules extracted from non 1-best

Rule	DepLM	BLEU
tree	tree	22.31
tree	forest	22.73*
forest	tree	22.80*
forest	forest	**23.12****

Table 3: BLEU scores on the Korean-Chinese test set.

parses are used by the decoder? Table 2 shows the number of rules filtered on the test set. We observe that the rule table size hardly increases. One possible reason is that we only keep the best dependency structure for each node. The last row shows that 16.86% of the rules used in 1-best derivations are extracted from non 1-best parses in the forests, indicating that some useful rules cannot be extracted from 1-best parses.

6.2 Results on the Korean-Chinese Task

To examine the efficacy of our approach on different language pairs, we carried out an experiment on Korean-Chinese translation. The training corpus contains about 8.2M Korean words and 7.3M Chinese words. The Chinese sentences were used to train a 5-gram language model as well as a 3-gram dependency language model. Both the development and test sets consist of 1,006 sentences with single reference. Table 3 shows the BLEU scores on the test set. Again, our forest-based approach achieves significant improvement over the baseline ($p < 0.01$).

6.3 Effect of K-best

We investigated the effect of different k-best structures for each node on translation quality (BLEU scores on the NIST 2005 set) and the rule table size (filtered for the tuning and test sets), as shown in Figure 5. To save time, we extracted rules just from the first 30K sentence pairs of the FBIS corpus. We trained a language model and depLMs on the English sentences. We used 10 different k: 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10. Obviously, the higher the k is, the more rules are extracted. When k=10, the number of rules used on the tuning and test sets was 1,299,290 and the BLEU score was 20.88. Generally, both the number of rules and the BLEU score went up with

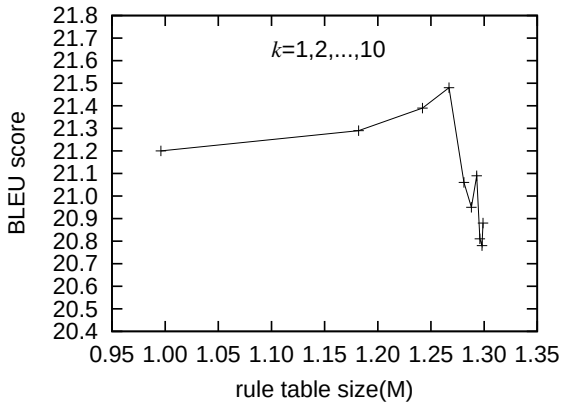


Figure 5: Effect of k-best on rule table size and translation quality.

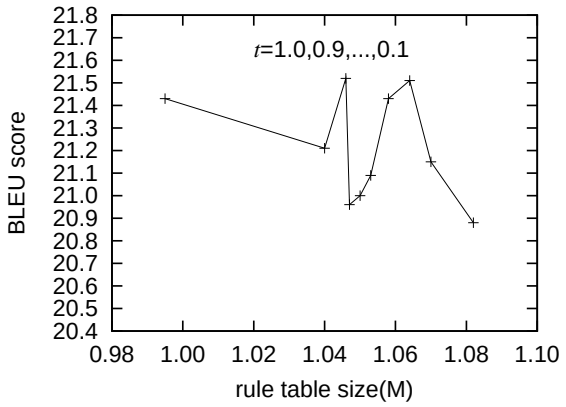


Figure 6: Effect of pruning threshold on rule table size and translation quality.

the increase of k. However, this trend did not hold within the range [4,10]. We conjecture that when retaining more dependency structures for each node, low quality structures would be introduced, resulting in much rules of low quality.

An interesting finding is that the rule table grew rapidly when k is in range [1,4], while gradually within the range [4,10]. One possible reason is that there are limited different dependency structures in the spans with a maximal length of 10, which the target side of rules cover.

6.4 Effect of Pruning Threshold

Figure 6 shows the effect of pruning threshold on translation quality and the rule table size. We retained 10-best dependency structures for each node in dependency forests. We used 10 different

pruning thresholds: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 and 1.0. Intuitively, the higher the pruning threshold is, the less rules are extracted. When t=0.1, the number of rules used on the tuning and test sets was 1,081,841 and the BLEU score was 20.68.

Lots of rules are pruned when the pruning threshold increases from 0.0 to 0.3 (around 20%). After pruning away these rules, we achieved 0.6 point improvement in BLEU. However, when we filtered more rules, the BLEU score went down.

Figures 5 and 6 show that using two parameters that have to be hand-tuned achieves a small improvement at the expense of an additional complexity. To simplify the approach, we only keep the best dependency structure for each node without pruning any rule.

7 Related Works

While Mi and Huang (2008) and we both use forests for rule extraction, there remain two major differences. Firstly, Mi and Huang (2008) use a packed forest, while we use a dependency forest. Packed forest is a natural weighted hypergraph (Klein and Manning, 2001; Huang and Chiang, 2005), for each hyperedge treats the corresponding PCFG rule probability as its weight. However, it is challenging to make dependency forest to be a weighted hypergraph because dependency parsers usually only output a score for each edge in a dependency tree rather than a hyperedge in a dependency forest. Secondly, The GHKM algorithm (Galley et al., 2004), which is originally developed for extracting tree-to-string rules from 1-best trees, has been successfully extended to packed forests recently (Mi and Huang, 2008). Unfortunately, the GHKM algorithm cannot be applied to extracting string-to-dependency rules from dependency forests, because the GHKM algorithm requires a complete subtree to exist in a rule while neither fixed nor floating dependency structures ensure that all dependants of a head are included.

8 Conclusion and Future Work

In this paper, we have proposed to use dependency forests instead of 1-best parses to extract string-to-dependency tree rules and train dependency language models. Our experiments show that our approach improves translation quality significantly over a state-of-the-art string-to-dependency system on various language pairs and test sets. We believe that dependency forest can also be used to improve the dependency treelet system (Quirk et al., 2005) that takes 1-best trees as input.

Acknowledgement

The authors were supported by SK Telecom C&I Business, and National Natural Science Foundation of China, Contracts 60736014 and 60903138. We thank the anonymous reviewers for their insightful comments. We are also grateful to Wenbin Jiang for his invaluable help in dependency forest.

References

Chiang, David. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, pages 201–228.

Ding, Yuan and Martha Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proceedings of ACL*.

Dyer, Christopher, Smaranda Muresan, and Philip Resnik. 2008. Generalizing word lattice translation. In *Proceedings of ACL*.

Galley, Michel, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of NAACL*.

Huang, Liang and David Chiang. 2005. Better k-best parsing. In *Proceedings of IWPT*.

Huang, Liang, Wenbin Jiang, and Qun Liu. 2009. Bilingually-constrained (monolingual) shift-reduce parsing. In *Proceedings of EMNLP*.

Klein, Dan and Christopher D. Manning. 2001. Parsing and hypergraphs. In *Proceedings of IWPT*.

Kneser, R. and H. Ney. 1995. Improved backing-off for m-gram language modeling. In *Proceedings of Acoustics, Speech, and Signal*.

Liu, Yang, Tian Xia, Xinyan Xiao, and Qun Liu. 2009. Weighted alignment matrices for statistical machine translation. In *Proceedings of EMNLP*.

Mi, Haitao and Liang Huang. 2008. Forest-based translation rule extraction. In *Proceedings of EMNLP*.

Och, Franz J. and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of ACL*.

Och, Franz J. and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of ACL*.

Och, Franz J. and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*.

Quirk, Chris and Simon Corston-Oliver. 2006. The impact of parsing quality on syntactically-informed statistical machine translation. In *Proceedings of EMNLP*.

Quirk, Chris, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: syntactically informed phrasal smt. In *Proceedings of ACL*.

Shen, Libin, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL*.

Stolcke, Andreas. 2002. Srilm - an extensible language modeling toolkit. In *Proceedings of ICSLP*.

Wang, Wei, Kevin Knight, and Daniel Marcu. 2007. Binarizing syntax trees to improve syntax-based machine translation accuracy. In *Proceedings of EMNLP*.

Large Scale Parallel Document Mining for Machine Translation

Jakob Uszkoreit Jay M. Ponte Ashok C. Popat Moshe Dubiner

Google, Inc.
{uszkoreit,ponte,popat,moshe}@google.com

Abstract

A distributed system is described that reliably mines parallel text from large corpora. The approach can be regarded as cross-language near-duplicate detection, enabled by an initial, low-quality batch translation. In contrast to other approaches which require specialized metadata, the system uses only the textual content of the documents. Results are presented for a corpus of over two billion web pages and for a large collection of digitized public-domain books.

1 Introduction

While the World Wide Web provides an abundance of readily available monolingual text, parallel data is still a comparatively scarce resource, yet plays a crucially important role in training statistical machine translation systems.

We describe an approach to mining document-aligned parallel text to be used as training data for a statistical machine translation system. Previous approaches have focused on rather homogeneous corpora and relied on metadata such as publication dates (Munteanu and Marcu, 2005; Munteanu and Marcu, 2006; Udupa et al., 2009; Do et al., 2009; Abdul-Rauf and Schwenk, 2009) or information about document structure (Resnik and Smith, 2003; Chen and Nie, 2000). In large and unstructured collections of documents such as the Web, however, metadata is often sparse or unreliable. Our approach, in contrast, scales computationally to very large and diverse collections of documents and does not require metadata. It is based solely on the textual contents of the input documents.

Casting the problem as one of cross-language near duplicate detection, we use a baseline machine translation system to translate all input documents into a single language. However, the words and phrases that are most discriminatory for the purposes of information retrieval and duplicate detection are the relatively rare ones, precisely those that are less likely to be translated well by the baseline translation system.

Our approach to circumvent this problem and to avoid the prohibitive quadratic computational complexity of the naive approach of performing a comparison of every possible pair of input documents is similar to previous work in near duplicate detection (Broder, 2000; Henzinger, 2006; Manber, 1994) and noisy data retrieval (Harding et al., 1997).

We use shingles consisting of word n-grams to construct relatively rare features from more common, in-vocabulary words. For each input document, we identify a comparatively small set of candidate pairings with documents sharing at least a certain number of such features. We then perform a more expensive comparison between each document and all documents in its candidate set using lower order n-gram features that would typically be too frequent to be used efficiently in forming candidate pairings, but provide a higher coverage of the scored document pairs. Another important aspect of our approach is that it can be implemented in a highly parallel way, as we describe in the following section.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1101–1109,
Beijing, August 2010

2 System Description

The input is a set of documents from diverse sources such as web pages and digitized books. In a first stage, all documents are independently translated into English using a baseline statistical machine translation system.

We then extract two different sets of n-grams from the translated documents: matching n-grams that are used to construct the candidate sets as well as scoring n-grams used only in the computation of a score for a given pair of documents. This stage generates two indexes: a *forward index* listing all extracted scoring n-grams, indexed by document; and an *inverted index* referencing all documents from which we extracted a given matching n-gram, indexed by n-grams. The inverted index is also used to accumulate global information about scoring n-grams, such as their document frequency, yet for scoring n-grams we do not accumulate a posting list of all documents in which they occur.

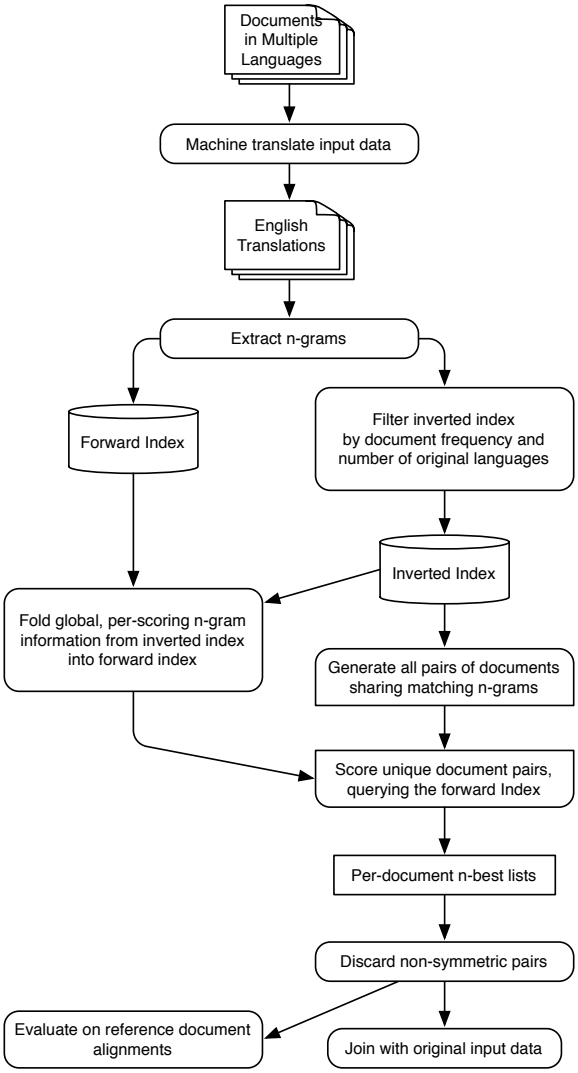


Figure 1: Architecture of the Parallel Text Mining System.

In the next step, the system generates all possible pairs of documents for each matching n-gram posting list in the inverted index. Since we keep only those pairs of documents that originated in different languages, we can discard posting lists from the inverted index that contain only a single document, i.e. those of singleton n-grams, or only documents in a single language.

Crucially, we further discard posting lists for matching n-grams whose frequency exceeds a certain threshold. When choosing a sufficiently large order for the matching n-grams, their long-tailed distribution causes only a small fraction of matching n-grams to be filtered out due to frequency, as we show empirically in Section 5. It is this filtering step that causes the overall runtime of the system to be linear in the size of the input data and allows the system to scale to very large document collections.

In parallel, global information about scoring n-grams accumulated in the inverted index that is required for pairwise scoring, such as their document frequency, is folded into the forward index by iterating over all forward index entries, requesting the respective per-feature quantities from the inverted index and storing them with each occurrence of a scoring n-gram in an updated forward index.

In the next stage, we compute pairwise scores for all candidate document pairs, accessing the forward index entry of each of the two scored documents to obtain the respective scoring n-grams. Document pairs with a score below a given threshold are discarded. For each input document, this results in one n-best list per language. In the last step we retain only those document pairs where each document is contained in the n-best list of the other document for its original language. Finally we perform a *join* of our identified translation pairs with the original text by making another

pass over the original, untranslated input data where the contents of document pairs with sufficiently high scores are then aggregated and output. Document pairings involving all languages are identified simultaneously. Each stage of the system fits well into the MapReduce programming model (Dean and Ghemawat, 2004). The general architecture is shown in Figure 1.

2.1 Pairwise Scoring

For scoring a pair of documents d and d', the forward index is queried for the entries for both documents. Let $F_d = \{f_1, f_2, ...f_n\}$ and $F_{d'} = \{f'_1, f'_2, ...f'_{n'}\}$ be the sets of scoring n-grams in the forward index entries of d and d', respectively. Let $\text{idf}(f) = \log \frac{|D|}{df(f)}$ be the inverse document frequency of a scoring n-gram f, where $|D|$ is the number of documents in the input corpus and $df(f)$ is the number documents from which we extracted the feature f. Interpreting F_d and $F_{d'}$ as incidence vectors in the vector space of n-grams and replacing each non-zero component f with $\text{idf}(f)$, we compute the score of the document pair as the inverse document frequency weighted cosine similarity of F_d and $F_{d'}$

$$\text{score}(d, d') = \frac{F_d \cdot F_{d'}}{||F_d|| \cdot ||F_{d'}||} \quad (1)$$

The per-document n-best lists are sorted according to this score and document pairs for which the score is below a threshold are discarded completely.

We do not use term frequency in the scoring metric. In preliminary experiments, incorporating the term frequency to yield basic *tf/idf* as well as using other information retrieval ranking functions incorporating term frequencies such as *BM25* (Robertson et al., 1995) resulted in a degradation of performance compared to the simpler scoring function described above. We believe this is due to the fact that, in contrast to the standard information retrieval setting, the overall length of our queries is on par with that of the documents in the collection.

The scoring is completely agnostic regarding the scoring n-grams' positions in the documents. Since especially for long documents such as books this may produce spurious matches, we apply an additional filter to remove document pairs for which the relative ordering of the matching scoring n-grams is very different. Together with each scoring n-gram we also extract its relative position in each document and store it in the forward index. When scoring a document pair, we compute the normalized permutation edit distance (Cormode et al., 2001) between the two sequences of overlapping n-grams sorted by their position in the respective document. If this distance exceeds a certain threshold, we discard the document pair.

2.2 Computational Complexity

By limiting the frequency of matching n-grams, the complexity becomes linear. Let the tunable parameter c be the maximum occurrence count for matching n-grams to be kept in the inverted index. Let m be the average number of matching n-grams extracted from a single document whose count is below c and D be the set of documents in the input corpus. Then the system generates up to $|D| \cdot m \cdot c$ candidate pairings. Scoring a given candidate document pair according to cosine similarity involves computing three dot-products between sparse vectors with one non-zero component per scoring n-gram extracted and not filtered from the respective document. Let s be the average number of such scoring n-grams per document, which is bounded by the average document length. Then the time complexity of the entire document alignment is in

$$O(|D| \cdot m \cdot c \cdot s) \quad (2)$$

and therefore linear in the number of input documents in the corpus and the average document size.

The space complexity is dominated by the size of the inverted and forward indexes, both of which are linear in the size of the input corpus.

2.3 Sentence-Level Alignment

Further filtering is performed on a per-sentence basis during per-document-pair sentence alignment of the mined text with a standard dynamic programming sentence alignment algorithm using sentence length and multilingual probabilistic dictionaries as features. Afterwards we crudely align

words within each pair of aligned source and target sentences. This crude alignment is used only to filter nonparallel sentences. Let S be the set of source words, T the set of target words and $S \times T$ the set of ordered pairs. Let the source sentence contain words $S_0 \subset S$ and the target sentence contain words $T_0 \subset T$. An alignment $A_0 \subset S_0 \times T_0$ will be scored by

$$\text{score}(A_0) = \sum_{(s,t) \in A_0} \ln \frac{p(s,t)}{p(s)\, p(t)} \qquad (3)$$

where the joint probabilities $p(s,t)$ and marginal probabilities $p(s)$, $p(t)$ are taken to be the respective empirical distributions (without smoothing) in an existing word aligned corpus. This is greedily maximized and the result is divided by its approximate expected value

$$\sum_{(s,t) \in S_0 \times T} \frac{p(s,t)}{p(s)} \ln \frac{p(s,t)}{p(s)\, p(t)} \qquad (4)$$

We discard sentence pairs for which the ratio between the actual and the expected score is less than $1/3$. We also drop sentence pairs for which both sides are identical, or a language detector declares them to be in the wrong language.

2.4 Baseline Translation System

To translate the input documents into English we use phrase-based statistical machine translation systems based on the log-linear formulation of the problem (Och and Ney, 2002).

We train the systems on the Europarl Corpus (Koehn, 2002), the DGT Multilingual Translation Memory (European Commission Directorate-General for Translation, 2007) and the United Nations ODS corpus (United Nations, 2006). Minimum error rate training (Macherey et al., 2008) under the BLEU criterion is used to optimize the feature function weights on development data consisting of the *nv-dev2007* and *news-dev2009* data sets provided by the organizers of the 2007 and 2009 WMT shared translation tasks[1]. We use a 4-gram language model trained on a variety of large monolingual corpora. The BLEU scores of our baseline translation system on the test sets from various WMT shared translation tasks are listed in Table 5. An empirical analysis of the impact of the baseline translation system quality on the data mining system is given in Section 6.3.

[1] available at `http://statmt.org`

3 Input Document Collections

We evaluate the parallel text mining system on two input data sets:

web A collection of 2.5 Billion general pages crawled from the Web, containing only pages in Czech, English, French, German, Hungarian and Spanish

books A collection of 1.5 Million public domain books digitized using an optical character recognition system. The collection consists primarily of English, French and fewer Spanish volumes

3.1 Reference Sets

We created reference sets of groups of documents in multiple languages which are true translations of one another for both the *web* and the *books* data set. Due to the presence of duplicates, each reference pairing can contain more than a single alternative translation per language. The *web* reference set was constructed by exploiting the systematic hyperlink structure of the web-site `http://america.gov/`, that links pages in one language to their respective translations into one or more other languages. The resulting reference set contains documents in Arabic, Chinese, English, French, Russian and Spanish, however, for most English pages there is only one translation into one of the other languages. Overall, the reference set contains 6,818 documents and 7,286 translation pairs.

The *books* reference set contains 30 manually aligned groups of translations covering a total of 103 volumes in English and French.

4 Evaluation Metrics

The fact that the system outputs pairs of documents and the presence of duplicate documents in the corpus motivate the use of modified versions of *precision* and *recall*.

Let C be a set of candidate parallel document pairs and let R be a possibly incomplete reference set of groups of parallel documents known to exist in the corpus. Consider the following two subsets of C:

- *Matching* pairs which are in some reference cluster.
- *Touching* pairs which are non-matching but have at least one document in some reference cluster.

We define

$$\text{Precision} = \frac{|C_{\text{Matching}}|}{|C_{\text{Matching}}| + |C_{\text{Touching}}|}$$

and

$$\text{Recall} = \frac{|C_{\text{Matching}}|}{|R|} \quad (5)$$

5 Parameter Selection

We conducted a series of small-scale experiments on only those documents contained in the *web* reference data set to empirically determine good settings for the tunable parameters of the text mining system. Among the most important parameters are the orders of the n-grams used for pairing documents as well as scoring them. Aside from the obvious impact on the quality of the output, these parameters have a very large influence on the overall computational performance of the system. The choice of the order of the extracted matching n-grams is mainly a trade-off between recall and efficiency. If the order is too large the system will miss valid pairs; if too small the the threshold on matching n-gram frequency will need to be increased.

Figure 2 shows the F1-scores obtained running only on the documents contained in the *web* reference set with different orders of matching and scoring n-grams. Figure 3 shows the corresponding number of pairwise comparisons made when using different orders of matching n-grams. While there is a drop of 0.01 in F1 score between using 2-grams and 5-grams as matching n-grams, this drop in quality seems to be well worth the 42-fold reduction in resulting pairwise comparisons.

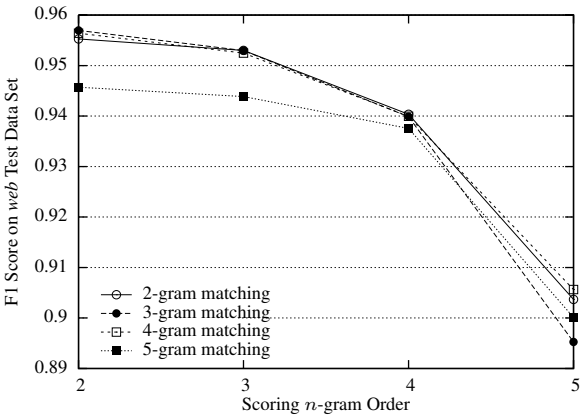


Figure 2: F1 scores on the *web* reference set for different scoring and matching n-gram orders.

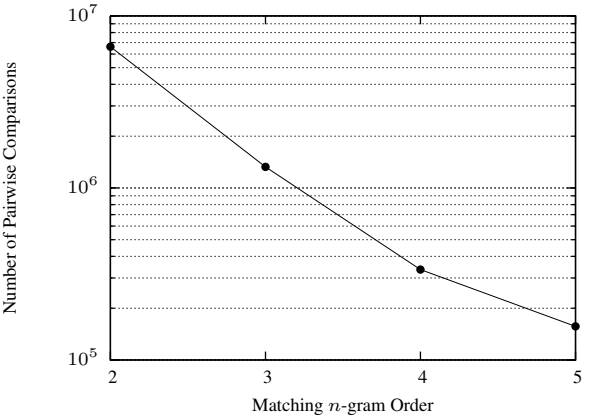


Figure 3: Number of pairwise comparisons made when using matching n-grams of different orders.

The largest portion of the loss in F1 score is incurred when increasing the matching n-gram order from 4 to 5, the reduction in pairwise comparisons, however, is still more than twofold.

Table 1 shows the precision and recall on the *web* reference set when running only on documents in the reference set using 5-grams as matching n-grams and bigrams for scoring for different values of the threshold on the cosine similarity score. In this setting as well as in large-scale experiments on both complete data sets described in section 6.1, a threshold of 0.1 yields the highest F1 score.

score threshold	0.06	0.10	0.12	0.16	0.20
precision	0.92	0.97	0.98	0.99	0.99
recall	0.91	0.91	0.90	0.89	0.83

Table 1: Precision and recall on the *web* reference set when running only on documents contained in the reference set.

6 Evaluation

We run the parallel text mining system on the *web* and *books* data sets using 5-grams for matching and bigrams for scoring. In both cases we discard matching n-grams which occurred in more than 50 documents and output only the highest scoring candidate for each document.

In case of the *web* data set, we extract every 5-gram as potential matching feature. For the *books* data set, however, we downsample the number of candidate matching 5-grams by extracting only those whose integer fingerprints under some hash function have four specific bits set, thus keeping on average only $1/16$ of the matching n-grams. Here, we also restrict the total number of matching n-grams extracted from any given document to 20,000. Scoring bigrams are dropped from the forward index if their document frequency exceeds 100,000, at which point their influence on the pairwise score would be negligible.

Running on the *web* data set, the system on average extracts 250 matching 5-grams per document, extracting a total of approximately 430 Billion distinct 5-grams. Of those, 78% are singletons and 21% only occur in a single language. Only approximately 0.8% of all matching n-grams are filtered due to having a document frequency higher than 50. The forward index initially contains more than 500 Billion bigram occurrences; after pruning out singletons and bigrams with a document frequency larger than 100,000, the number of indexed scoring feature occurrences is reduced to 40%. During scoring, approximately 50 Billion pairwise comparisons are performed.

In total the n-gram extraction, document scoring and subsequent filtering takes less than 24 hours on a cluster of 2,000 state-of-the-art CPUs.

The number of words after sentence-level filtering and alignment that the parallel text mining system extracted for the different languages from each dataset are listed in Table 2.

	baseline	*books*	*web*
Czech	27.5 M	0	271.9 M
French	479.8 M	228.5 M	4,914.3 M
German	54.2 M	0	3,787.6 M
Hungarian	26.9 M	0	198.9 M
Spanish	441.0 M	15.0 M	4,846.8 M

Table 2: The number of words per language in the baseline training corpora and extracted from the two different data sets.

score threshold	0.06	0.10	0.12	0.16	0.20
precision	0.88	0.93	0.95	0.97	0.97
recall	0.68	0.65	0.63	0.52	0.38

Table 3: Precision and recall on the reference set when running on the complete *web* data set with different score thresholds.

score threshold	0.06	0.10	0.12	0.16	0.20
precision	0.95	1.00	1.00	1.00	1.00
recall	0.71	0.71	0.71	0.48	0.38

Table 4: Precision and recall on the reference set when running on the complete *books* data set with different score thresholds.

6.1 Precision and Recall

Tables 3 and 4 show precision and recall on the respective reference sets for the *web* and the *books* input data sets. While the text mining system maintains a very high precision, recall drops significantly compared to running only on the documents in the reference set. One reason for this behavior is that the number of n-grams in the test data set which are sufficiently rare to be used as queries drops with increasing amounts of input data and in particular short documents which only share a small number of matching n-grams anyway, may happen to only share matching n-grams with a too high document frequency. Further analysis shows that another, more significant factor is the existence of multiple, possibly partial translations and near-duplicate documents which cause symmetrization to discard valid document pairs because each document in the pair is determined by the document pair score to be more similar to a different translation of a near-duplicate or sub-

Language Pair	Training Data	WMT 2007 news commentary	WMT 2008 news	WMT 2009 news
Czech English	baseline	21.59	14.59	16.46
	web	29.26 (+7.67)	20.16 (+5.57)	23.25 (+6.76)
German English	baseline	27.99	20.34	20.03
	web	32.35 (+4.36)	23.22 (+2.88)	23.35 (+3.32)
Hungarian English	baseline	-	10.21	11.02
	web	-	12.92 (+2.71)	14.68 (+3.66)
French English	baseline	34.26	22.14	26.39
	books	34.73 (+0.47)	22.39 (+0.25)	27.15 (+0.76)
	web	36.65 (+2.39)	23.22 (+1.08)	28.34 (+1.95)
Spanish English	baseline	43.67	24.15	26.88
	books	44.07 (+0.40)	24.32 (+0.17)	27.16 (+0.28)
	web	46.21 (+2.54)	25.52 (+1.37)	28.50 (+1.62)
English Czech	baseline	14.78	12.45	11.62
	web	20.65 (+5.86)	18.70 (+6.25)	16.60 (+4.98)
English German	baseline	19.89	14.67	14.31
	web	23.49 (+3.60)	16.78 (+2.11)	16.96 (+2.65)
English Hungarian	baseline	-	07.93	08.52
	web	-	10.16 (+2.23)	11.42 (+2.90)
English French	baseline	31.59	22.29	25.14
	books	31.92 (+0.33)	22.42 (+0.13)	25.46 (+0.32)
	web	34.35 (+2.76)	23.56 (+1.27)	27.05 (+1.91)
English Spanish	baseline	42.05	24.65	25.85
	books	42.05	24.79 (+0.14)	26.07 (+0.22)
	web	45.21 (+3.16)	26.46 (+1.81)	27.79 (+1.94)

Table 5: BLEU scores of the translation systems trained on the automatically mined parallel corpora and the baseline training data.

set of the document. This problem seems to affect news articles in particular where there are often multiple different translations of large subsets of the same or slightly changed versions of the article.

6.2 Translation Quality

Arabic English	NIST 2006	NIST 2008
Baseline (UN ODS)	44.31	42.79
Munteanu and Marcu	45.13	43.86
Present work	44.72	43.64
Chinese English	NIST 2006	NIST 2008
Baseline (UN ODS)	25.71	19.79
Munteanu and Marcu	28.11	21.69
Present work	28.08	22.02

Table 6: BLEU scores of the Chinese and Arabic to English translation systems trained on the baseline UN ODS corpus and after adding either the Munteanu and Marcu corpora or the training data mined using the presented approach.

We trained a phrase-based translation system on the mined parallel data sets and evaluated it on translation tasks for the language pairs Czech, French, German, Hungarian and Spanish to and from English, measuring translation quality with the BLEU score (Papineni et al., 2002). The translation tasks evaluated are the WMT 2007 news commentary test set as well the WMT 2008 and 2009 news test sets.

The parallel data for this experiment was mined using the general settings described in the previous section and a threshold of 0.1 on the pairwise score. We ensure that the test data is not included in the training data by filtering out all sentences from the training data that share more than 30% of their 6-grams with any sentence from one of the test corpora.

Table 5 shows the BLEU scores of the different translation systems. The consistent and significant improvements in BLEU score demonstrate the usefulness of the mined document pairs in training a translation system.

Even though the presented approach works on a less granular level than the sentence-level approach of Munteanu and Marcu (2005), we compare results on the same input data[2] used by those authors to automatically generate the

[2]LDC corpora LDC2005T12, LDC2005T14 and LDC2006T02, the second editions of the Arabic, Chinese and English Gigaword corpora.

Sampling Rate	WMT 2007 news commentary			WMT 2008 news			WMT 2009 news		
	degraded	Cz→En	En→Cz	degraded	Cz→En	En→Cz	degraded	Cz→En	En→Cz
1.0	21.59	29.26	20.65	14.59	20.16	18.70	16.46	23.25	16.60
0.5	20.12	29.16	20.55	13.65	20.16	18.71	15.44	23.16	16.56
0.25	18.59	29.09	20.61	12.79	20.09	18.58	14.35	23.18	16.50
0.125	16.69	29.10	20.39	11.87	20.07	18.48	13.05	23.06	16.53
0.0625	14.72	29.04	20.44	10.87	20.06	18.49	11.62	23.11	16.44
0.0312	12.60	28.75	20.28	09.71	19.97	18.45	10.43	23.04	16.41

Table 7: BLEU scores of the degraded Czech to English baseline systems used for translating Czech documents from the *web* data set as well as those of Czech to and from English systems trained on data mined using translations of varying quality created by sampling from the training data.

Arabic English and Chinese English sentence-aligned parallel LDC corpora LDC2007T08 and LDC2007T09. We trained Arabic and Chinese English baseline systems on the United Nations ODS corpus (United Nations, 2006); we also use these to translate the non-English portions of the input data to English. We then evaluate the effects of also training on either the LDC2007T08 and LDC2007T09 corpora or the parallel documents mined by our approach in addition to the United Nations ODS corpus on the NIST 2006 and 2008 MT evaluation test sets. The results are presented in Table 6.

The approach proposed in (Munteanu and Marcu, 2005) relies critically on the existence of publication dates in order to be computationally feasible, yet it still scales superlinearly in the amount of input data. It could therefore not easily be applied to much larger and less structured input data collections. While our approach neither uses metadata nor operates on the sentence level, in all but one of the tasks, the system trained on the data mined using our approach performs similarly or slightly better.

6.3 Impact of Baseline Translation Quality

In order to evaluate the impact of the translation quality of the baseline system on the quality of the mined document pairs, we trained artificially degraded Czech to English translation systems by sampling from the baseline training data at decreasing rates. We translate the Czech subset of the *web* document collection into English with each of the degraded systems and apply the parallel data mining system in the same configuration.

Table 7 shows the BLEU scores of the degraded baseline systems and those resulting from adding the different mined data sets to the non-degraded Czech English and English Czech systems. Degrading the input data translation quality by up to 8.9% BLEU results in a consistent but only comparatively small decrease of less than 0.6% BLEU in the scores obtained when training on the mined document pairs. This does not only show that the impact of variations of the baseline system quality on the data mining system is limited, but also that the data mining system will already work with a rather low quality baseline system.

7 Conclusion

We presented a scalable approach to mining parallel text from collections of billions of documents with high precision. The system makes few assumptions about the input documents. We demonstrated that it works well on different types of data: a large collection of web pages and a collection of digitized books. We further showed that the produced parallel corpora can significantly improve the quality of a state-of-the-art statistical machine translation system.

8 Acknowledgments

We thank the anonymous reviewers for their insightful comments.

References

Abdul-Rauf, Sadaf and Holger Schwenk. 2009. On the use of comparable corpora to improve SMT performance. In *EACL*, pages 16–23.

Broder, Andrei Z. 2000. Identifying and filtering near-duplicate documents. In *COM '00: Proceedings of the 11th Annual Symposium on Combinatorial Pat-*

tern Matching, pages 1–10, London, UK. Springer-Verlag.

Chen, Jiang and Jian-Yun Nie. 2000. Parallel web text mining for cross-language IR. In *In In Proc. of RIAO*, pages 62–77.

Cormode, Graham, S. Muthukrishnan, and Süleyman Cenk Sahinalp. 2001. Permutation editing and matching via embeddings. In *ICALP '01: Proceedings of the 28th International Colloquium on Automata, Languages and Programming,*, pages 481–492, London, UK. Springer-Verlag.

Dean, Jeffrey and Sanjay Ghemawat. 2004. MapReduce: Simplified data processing on large clusters. In *Proceedings of the Sixth Symposium on Operating System Design and Implementation (OSDI-04)*, San Francisco, CA, USA.

Do, Thi-Ngoc-Diep, Viet-Bac Le, Brigitte Bigi, Laurent Besacier Eric, and Castelli. 2009. Mining a comparable text corpus for a Vietnamese - French statistical machine translation system. In *Proceedings of the 4th EACL Workshop on Statistical Machine Translation*, pages 165–172, Athens, Greece, March.

European Commission Directorate-General for Translation. 2007. DGT-TM parallel corpus. `http://langtech.jrc.it/DGT-TM.html`.

Harding, Stephen M., W. Bruce Croft, and C. Weir. 1997. Probabilistic retrieval of OCR degraded text using n-grams. In *ECDL '97: Proceedings of the First European Conference on Research and Advanced Technology for Digital Libraries*, pages 345–359, London, UK. Springer-Verlag.

Henzinger, Monika. 2006. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 284–291, New York, NY, USA. ACM.

Koehn, Philipp. 2002. Europarl: A multilingual corpus for evaluation of machine translation. Draft.

Macherey, Wolfgang, Franz Och, Ignacio Thayer, and Jakob Uszkoreit. 2008. Lattice-based minimum error rate training for statistical machine translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 725–734, Honolulu, Hi, October. Association for Computational Linguistics.

Manber, Udi. 1994. Finding similar files in a large file system. In *Proceedings of the USENIX Winter 1994 Technical Conferenc*.

Munteanu, Dragos Stefan and Daniel Marcu. 2005. Improving machine translation performance by exploiting non-parallel corpora. *Comput. Linguist.*, 31(4):477–504.

Munteanu, Dragos Stefan and Daniel Marcu. 2006. Extracting parallel sub-sentential fragments from non-parallel corpora. In *ACL*.

Och, Franz Josef and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 295–302, Philadelphia, PA, USA.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318, Philadelphia, PA, USA.

Resnik, Philip and Noah A. Smith. 2003. The web as a parallel corpus. *Computational Linguistics*, 29:349–380.

Robertson, S E, S Walker, S Jones, M M Hancock-Beaulieu, and M Gatford. 1995. Okapi at TREC–3. In *Proceedings of the Third Text REtrieval Conference (TREC-3)*.

Udupa, Raghavendra, K. Saravanan, A. Kumaran, and Jagadeesh Jagarlamudi. 2009. Mint: A method for effective and scalable mining of named entity transliterations from large comparable corpora. In *EACL*, pages 799–807.

United Nations. 2006. ODS UN parallel corpus. `http://ods.un.org/`.

Hungarian Corpus of Light Verb Constructions

Veronika Vincze
University of Szeged
Department of Informatics
vinczev@inf.u-szeged.hu

János Csirik
Hungarian Academy of Sciences
Research Group on Artificial Intelligence
csirik@inf.u-szeged.hu

Abstract

The precise identification of light verb constructions is crucial for the successful functioning of several NLP applications. In order to facilitate the development of an algorithm that is capable of recognizing them, a manually annotated corpus of light verb constructions has been built for Hungarian. Basic annotation guidelines and statistical data on the corpus are also presented in the paper. It is also shown how applications in the fields of machine translation and information extraction can make use of such a corpus and an algorithm.

1 Introduction

In this paper, we report a corpus containing light verb constructions in Hungarian. These expressions are neither productive nor idiomatic and their meaning is not totally compositional (the noun is usually taken in one of its literal senses but the verb usually loses its original sense to some extent), as it can be seen in the examples from different languages shown below. Since their meaning is the same, only literal translations are provided:

- English: *to give a lecture*, *to come into bloom*, *the problem lies (in)*
- German: *halten eine Vorlesung* to hold a presentation, *in Blüte stehen* in bloom to stand, *das Problem liegt (in)* the problem lies (in)
- French: *faire une présentation* to make a presentation, *être en fleur* to be in bloom, *le problème réside (dans)* the problem resides (in)
- Hungarian: *előadást tart* presentation-ACC holds, *virágba borul* bloom-ILL falls, *probléma rejlik (vmiben)* problem hides (in sg)

Several terms like *complex verb structures*, *support verb constructions* or *light verb constructions* have been used[1] for these constructions in the literature (Langer, 2004). In this paper, the term *light verb constructions* will be employed.

The structure of the paper is as follows. First, the importance of the special NLP treatment of light verb constructions is emphasized in section 2. The precise identification of such constructions is crucial for the successful functioning of NLP applications, thus, it is argued that an algorithm is needed to automatically recognize them (section 4). In order to facilitate the development of such an algorithm, a corpus of light verb constructions has been built for Hungarian, which is presented together with statistical data in section 5. Finally, it is shown how NLP applications in the fields of machine translation and information extraction can profit from the implementation of an algorithm capable of identifying light verb constructions (section 6).

2 Light verb constructions in NLP

In natural language processing, one of the most challenging tasks is the proper treatment of col-

[1] There might be slight theoretical differences in the usage of these terms – e.g. semantically empty support verbs are called *light verbs* in e.g. Meyers et al. (2004a), that is, the term *support verb* is a hypernym of *light verb*. However, these differences are not analyzed in detail in this paper.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1110–1118,
Beijing, August 2010

locations, which term comprises light verb constructions as well. Every multiword expression is considered to be a collocation if its members often co-occur and its form is fixed to some extent (Siepmann, 2005; Siepmann, 2006; Sag et al., 2001; Oravecz et al., 2004; Váradi, 2006). Collocations are frequent in language use and they usually exhibit unique behaviour, thus, they often pose a problem to NLP systems.

Light verb constructions deserve special attention in NLP applications for several reasons. First, their meaning is not totally compositional, that is, it cannot be computed on the basis of the meanings of the parts of the collocation and the way they are related to each other. Thus, the result of translating the parts of the collocation can hardly be considered as the proper translation of the original expression. Second, light verb constructions (e.g. *make a mistake*) often share their syntactic pattern with other constructions such as literal verb + noun combinations (e.g. *make a cake*) or idioms (e.g. *make a meal*), thus, their identification cannot be based on solely syntactic patterns. Third, since the syntactic and the semantic head of the construction are not the same – the syntactic head being the verb and the semantic head being the noun –, they require special treatment when parsing. It can be argued that they form a complex verb similarly to phrasal or prepositional verbs (as reflected in the term complex verb structures). Thus, it is advisable to indicate their special syntacto-semantic relationship: in dependency grammars, the new role QUASI-ARGUMENT might be proposed for this purpose.

3 Related work

Light verb constructions – as a subtype of multiword expressions – have been paid special attention in NLP literature. Sag et al. (2001) classify them as a subtype of lexicalized phrases and flexible expressions. They are usually distinguished from productive or literal verb + noun constructions on the one hand and idiomatic verb + noun expressions on the other hand: e.g. Fazly and Stevenson (2007) use statistical measures in order to classify subtypes of verb + noun combinations and Diab and Bhutada (2009) developed a chunking method for classifying multiword expressions.

Identifying multiword expressions in general and light verb constructions in particular is not unequivocal since constructions with similar syntactic structure (e.g. verb + noun combinations) can belong to different subclasses on the productivity scale (i.e. productive combinations, light verb constructions and idioms). That is why well-designed and tagged corpora of multiword expressions are invaluable resources for training and testing algorithms that are able to identify multiword expressions. For instance, Grégoire (2007) describes the design and implementation of a lexicon of Dutch multiword expressions. Focusing on multiword verbs, Kaalep and Muischnek (2006; 2008) present an Estonian database and a corpus and Krenn (2008) describes a database of German PP-verb combinations. The Prague Dependency Treebank also contains annotation for light verb constructions (Cinková and Kolářová, 2005) and NomBank (Meyers et al., 2004b) provides the argument structure of common nouns, paying attention to those occurring in support verb constructions as well. On the other hand, Zarrieß and Kuhn (2009) make use of translational correspondences when identifying multiword expressions (among them, light verb constructions). A further example of corpus-based identification of light verb constructions in English is described in Tan et al. (2006).

Light verb constructions are considered to be semi-productive, that is, certain verbs tend to co-occur with nouns belonging to a given semantic class. A statistical method is applied to measure the acceptability of possible light verb constructions in Stevenson et al. (2004), which correlates reasonably well with human judgments.

4 Identifying light verb constructions

A database of light verb constructions and an annotated corpus might be of great help in the automatic recognition of light verb constructions. They can serve as a training database when implementing an algorithm for identifying those constructions.

The recognition of light verb constructions cannot be solely based on syntactic patterns for other (productive or idiomatic) combinations may exhibit the same verb + noun scheme (see section

2). However, in agglutinative languages such as Hungarian, nouns can have several grammatical cases, some of which typically occur in a light verb construction when paired with a certain verb. For instance, the verb *hoz* 'bring' is a transitive verb, that is, it usually occurs with a noun in the accusative case. On the other hand, when it is preceded or followed by a noun in the sublative or illative case (the typical position of the noun in Hungarian light verb constructions being right before or after the verb[2]), it is most likely a light verb construction. To illustrate this, we offer some examples:

vizet hoz

water-ACC bring

'to bring some water'

zavarba hoz

trouble-ILL bring

'to embarrass'

The first one is a productive combination (with the noun being in the accusative form) while the second one is a light verb construction. Note that the light verb construction also has got an argument in the accusative case (syntactically speaking, a direct object complement) as in:

Ez a megjegyzés mindenkit zavarba hozott.

this the remark everyone-ACC trouble-ILL bring-PAST-3SG

'This remark embarrassed everybody.'

Thus, the presence of an argument in the accusative does not imply that the noun + verb combination is a light verb construction. On the other hand, the presence of a noun in the illative or sublative case immediately preceding or following the verb strongly suggests that a light verb instance of *hoz* is under investigation.

Most light verb constructions have a verbal counterpart derived from the same stem as the noun, which entails that it is mostly deverbal nouns that occur in light verb constructions (as in *make/take a* ***decision*** compared to ***decide*** or ***döntést*** *hoz* vs. ***dönt*** in Hungarian). The identification of such nouns is possible with the help of a morphosyntactic parser that is able to treat derivation as well (e.g. hunmorph for Hungarian (Trón et al., 2005)), and the combination of a possible light verb and a deverbal noun typically results in a light verb construction.

Thus, an algorithm that makes use of morphosyntactic and derivational information and previously given lists can be constructed to identify light verb constructions in texts. It is important that the identification of light verb constructions precedes syntactic parsing, for the noun and the verb in the construction form one complex predicate, which has its effects on parsing: other arguments belong not solely to the verb but to the complex predicate.

To the best of our knowledge, there are no corpora of light verb constructions available for Hungarian. That is why we decided to build such a corpus. The corpus is described in detail in section 5. On the basis of the corpus developed, we plan to design an algorithm to automatically identify light verb constructions in Hungarian.

5 The corpus

In order to facilitate the extraction and the NLP treatment of Hungarian light verb constructions, we decided to build a corpus in which light verb constructions are annotated. The Szeged Treebank (Csendes et al., 2005) – a database in which words are morphosyntactically tagged and sentences are syntactically parsed – constitutes the basis for the annotation. We first selected the subcorpora containing business news, newspaper texts and legal texts for annotation since light verb constructions are considered to frequently occur in these domains (see B. Kovács (1999)). However, we plan to extend the annotation to other subcorpora as well (e.g. literary texts) in a later phase. Statistical data on the annotated subcorpora can be seen in Table 1.

5.1 Types of light verb constructions

As Hungarian is an agglutinative language, light verb constructions may occur in various forms.

[2]In a neutral sentence, the noun is right before the verb, in a sentence containing focus, it is right after the verb.

	sentences	words
business news	9574	186030
newspapers	10210	182172
legal texts	9278	220069
total	29062	582871

Table 1: Number of sentences and words in the annotated subcorpora

For instance, the verbal component may be inflected for tense, mood, person, number, etc. However, these inflectional differences can be easily resolved by a lemmatizer. On the other hand, besides the prototypical noun + verb combination, light verb constructions may be present in different syntactic structures, that is, in participles and infinitives and they can also undergo nominalization. These types are all annotated in the corpus texts since they also occur relatively frequently (see statistical data in 5.3). All annotated types are illustrated below.

- **Noun + verb combination <verb>**

  bejelentést tesz

  announcement-ACC makes

  'to make an announcement'

- **Participles <part>**

  - Present participle
    életbe lépő (intézkedés)
    life-ILL stepping (instruction)
    '(an instruction) taking effect'
  - Past participle
    csődbe ment (cég)
    bankrupt-ILL gone (firm)
    '(a firm) that went bankrupt'
  - Future participle
    fontolóra veendő (ajánlat)
    consideration-SUB to be taken (offer)
    '(an offer) that is to be taken into consideration'
  - Infinitive
    forgalomba hozni
    circulation-ILL bring-INF
    'to put into circulation'

- **Nominalization <nom>**

  bérbe vétel

  rent-ILL taking

  'hiring'

Split light verb constructions, where the noun and the verb are not adjacent, are also annotated and tagged. In this way, their identification becomes possible and the database can be used for training an algorithm that automatically recognizes (split) light verb constructions.

5.2 Annotation principles

Corpus texts contain single annotation, i.e. one annotator worked on each text. Light verb constructions can be found in between XML tags <FX> </FX>. In order to decide whether a noun + verb combination is a light verb construction or not, annotators were suggested to make use of a test battery developed for identifying Hungarian light verb constructions (Vincze, 2008).

The annotation process was carried out manually on the syntactically annotated version of the Szeged Treebank, thus, phrase boundaries were also taken into consideration when marking light verb constructions. Since the outmost boundary of the nominal component was considered to be part of the light verb construction, in several cases adjectives and other modifiers of the nominal head are also included in the construction, e.g.:

*<FX>**nyilvános** ajánlatot tesz</FX>*

public offer-ACC make

'to make a public offer'

In the case of participles, NP arguments may be also included (although in English, the same argument is expressed by a PP):

*<FX>**Nyíregyházán** tartott ülésén</FX>*

Nyíregyháza-SUP hold-PPT session-3SGPOSS-SUP

'at its session held in Nyíregyháza'

Constructions with a nominal component in the accusative case can be nominalized in two ways in Hungarian, as in:

szerződést köt

contract-ACC bind

'to make a contract'

<FX>szerződéskötés</FX>

contract+bind-GERUND

'making a contract'

<FX>adásvételi szerződések megkötése</FX>

sale contract-PL PREVERB-bind-GERUND-3SGPOSS

'making of sales contracts'

Both types are annotated in the corpus.

Besides the prototypical occurrences of light verb constructions (i.e. a bare common noun + verb[3]), other instances were also annotated in the corpus. For instance, the noun might be accompanied by an article or a modifier (recall that phrase boundaries were considered during annotation) or – for word order requirements – the noun follows the verb as in:

Ő hozta a jó döntést.

he bring-PAST-3SG-OBJ the good decision-ACC

'It was him who made the good decision.'

For the above reasons, a single light verb construction manifests in several different forms in the corpus. However, each occurrence is manually paired with its prototypical (i.e. bare noun + verb) form in a separate list, which is available at the corpus website.

5.3 Statistics on corpus data

The database contains 3826 occurrences of 658 light verb constructions altogether in 29062 sentences. Thus, a specific light verb construction occurs 5.8 times in the corpus on average. However, the participle form *irányadó* occurs in 607 instances (e.g. in *irányadó kamat* 'prime rate') due to the topic of the business news subcorpus, which may distort the percentage rates. For this reason, statistical data in Table 2 are shown the occurrences of *irányadó* excluded.

	verb	part	nom	split	total
business news	565 58.6%	270 28%	90 9.3%	40 4.1%	965 25.2%
newspapers	458 59.3%	192 24.9%	55 7.1%	67 8.7%	772 20.2%
legal texts	640 30.7%	504 24.1%	709 33.9%	236 11.3%	2089 54.6%
total	1663 43.5%	966 25.2%	854 22.3%	236 9%	3826 100%

Table 2: Subtypes of light verb constructions in the corpus

It is revealed that although it is verbal occurrences that are most frequent, the percentage rate of participles is also relatively high. The number of nominalized or split constructions is considerably lower (except for the law subcorpus, where their number is quite high), however, those together with participles are responsible for about 55% of the data, which indicates the importance of their being annotated as well.

As for the general frequency of light verb constructions in texts, we compared the number of verb + argument relations found in the Szeged Dependency Treebank (Vincze et al., 2010) where the argument was a common noun to that of light verb constructions. It has turned out that about 13% of verb + argument relations consist of light verb constructions. This again emphasizes that they should be paid attention to, especially in the legal domain (where this rate is as high as 36.8%). Statistical data are shown in Table 3.

	V + argument	LVC
business news	9524	624 (6.6%)
newspapers	3637	539 (14.8%)
legal texts	2143	889 (36.8%)
total	15574	2052 (13.2%)

Table 3: Verb + argument relations and light verb constructions

The corpus is publicly available for re-

[3]As opposed to other languages where prototypical light verb constructions consist of a verb + a noun in accusative or a verb + a prepositional phrase (see e.g. Krenn (2008)), in Hungarian, postpositional phrases rarely occur within a light verb construction. However, annotators were told to annotate such cases as well.

search and/or educational purposes at www.inf.u-szeged.hu/rgai/nlp.

6 The usability of the corpus

As emphasized earlier, the proper treatment of light verb constructions is of primary importance in NLP applications. In order to achieve this, their identification is essential. The corpus created can function as the training database for the implementation of an algorithm capable of recognizing light verb constructions, which we plan to develop in the near future. In the following, the ways machine translation and information extraction can profit from such a corpus and algorithm are shortly presented.

6.1 Light verb constructions and machine translation

When translating collocations, translation programs face two main problems. On the one hand, parts of the collocation do not always occur next to each other in the sentence (split collocations). In this case, the computer must first recognize that the parts of the collocation form one unit (Oravecz et al., 2004), for which the multiword context of the given word must be considered. On the other hand, the lack (or lower degree) of compositionality blocks the possibility of word-by-word translation (Siepmann, 2005; Siepmann, 2006). However, a (more or less) compositional account of light verb constructions is required for successful translation (Dura and Gawrońska, 2005).

To overcome these problems, a reliable method is needed to assure that the nominal and verbal parts of the construction be matched. This requires an algorithm that can identify light verb constructions. In our corpus, split light verb constructions are also annotated, thus, it is possible to train the algorithm to recognize them as well: the problem of split collocations can be eliminated in this way.

A comprehensive list of light verb constructions can enhance the quality of machine translation – if such lists are available for both the source and the target language. Annotated corpora (especially and most desirably, parallel corpora) and explanatory-combinatorial dictionaries[4] are possible sources of such lists. Since in foreign language equivalents of light verb constructions, the nominal components are usually literal translations of each other (Vincze, 2009), by collating the corresponding noun entries in these lists the foreign language variant of the given light verb construction can easily be found. On the other hand, in order to improve the building of such lists, we plan to annotate light verb constructions in a subcorpus of SzegedParalell, a Hungarian-English manually aligned parallel corpus (Tóth et al., 2008).

6.2 Light verb constructions and information extraction

Information extraction (IE) seeks to process large amounts of unstructured text, in other words, to collect relevant items of information and to classify them. Even though humans usually overperform computers in complex information processing tasks, computers also have some obvious advantages due to their capacity of processing and their precision in performing well-defined tasks.

For several IE applications (e.g. relationship extraction) it is essential to identify phrases in a clause and to determine their grammatical role (subject, object, verb) as well. This can be carried out by a syntactic parser and is a relatively simple task. However, the identification of the syntactic status of the nominal component is more complex in the case of light verb constructions for it is a quasi-argument of the verb not to be confused with other arguments (Alonso Ramos, 1998). Thus, the parser should recognize the special status of the quasi-argument and treat it in a specific way as in the following sentences, one of which contains a light verb construction while the other one a verbal counterpart of the construction:

*Pete **made a decision** on his future.*

*Pete **decided** on his future.*

[4]Explanatory combinatorial dictionaries are essential for relation descriptions (up to the present, only fractions of the dictionary have been completed for Russian (Mel'čuk and Žolkovskij, 1984) and for French (see Mel'čuk et al. (1984 1999)), besides, trial entries have been written in Polish, English and German that contain the relations of a certain lexical unit to other lexemes given by means of lexical functions (see e.g. Mel'čuk et al. (1995)). These dictionaries indicate light verb constructions within the entry of the nominal component.

In the sentence with the verbal counterpart, the event of deciding involves two arguments: *he* and *his future*. In the sentence with the light verb construction, the same arguments can be found, however, it is unresolved whether they are the arguments of the verb (*made*) or the nominal component (*decision*). If a precise syntactic analysis is needed, it is crucial to know which argument belongs to which governor. Nevertheless, it is still debated if syntactic arguments should be divided between the nominal component and the verb (see Meyers et al. (2004a) on argument sharing) and if yes, how (Alonso Ramos, 2007).

For the purpose of information extraction, such a detailed analysis is unnecessary and in general terms, the nominal component can be seen as part of the verb, that is, they form a complex verb similarly to phrasal or prepositional verbs and this complex verb is considered to be the governor of arguments. Thus, the following data can be yielded by the IE algorithm: there is an event of **decision-making**, **Pete** is its subject and it is about **his future** (and not an event of **making** with the arguments **decision**, **Pete** and **his future**). Again, the precise identification of light verb constructions can highly improve the performance of parsers in recognizing relations between the complex verb and its arguments.

7 Conclusion

In this paper, we have presented the development of a corpus of Hungarian light verb constructions. Basic annotation guidelines and statistical data have also been included. The annotated corpus can serve as a training database for implementing an algorithm that aims at identifying light verb constructions. Several NLP applications in the fields of e.g. machine translation and information extraction may profit from the successful integration of such an algorithm into the system, which we plan to develop in the near future.

Acknowledgements

This work was supported in part by the National Office for Research and Technology of the Hungarian government within the framework of the project MASZEKER.

The authors wish to thank György Szarvas for his help in developing the annotation tool and Richárd Farkas for his valuable comments on an earlier draft of this paper.

References

Alonso Ramos, Margarita. 1998. *Etude sémantico-syntaxique des constructions à verbe support*. Ph.D. thesis, Université de Montréal, Montreal, Canada.

Alonso Ramos, Margarita. 2007. Towards the Synthesis of Support Verb Constructions. In Wanner, Leo, editor, *Selected Lexical and Grammatical Issues in the Meaning-Text Theory. In Honour of Igor Mel'čuk*, pages 97–138, Amsterdam / Philadelphia. Benjamins.

B. Kovács, Mária. 1999. A funkcióigés szerkezetek a jogi szaknyelvben [Light verb constructions in the legal terminology]. *Magyar Nyelvőr*, 123(4):388–394.

Cinková, Silvie and Veronika Kolářová. 2005. Nouns as Components of Support Verb Constructions in the Prague Dependency Treebank. In Šimková, Mária, editor, *Insight into Slovak and Czech Corpus Linguistics*, pages 113–139. Veda Bratislava, Slovakia.

Csendes, Dóra, János Csirik, Tibor Gyimóthy, and András Kocsor. 2005. The Szeged TreeBank. In Matousek, Václav, Pavel Mautner, and Tomás Pavelka, editors, *Proceedings of the 8th International Conference on Text, Speech and Dialogue, TSD 2005*, Lecture Notes in Computer Science, pages 123–132, Berlin / Heidelberg, September. Springer.

Diab, Mona and Pravin Bhutada. 2009. Verb Noun Construction MWE Token Classification. In *Proceedings of the Workshop on Multiword Expressions: Identification, Interpretation, Disambiguation and Applications*, pages 17–22, Singapore, August. Association for Computational Linguistics.

Dura, Elżbieta and Barbara Gawrońska. 2005. Towards Automatic Translation of Support Verbs Constructions: the Case of Polish robic/zrobic and Swedish göra. In *Proceedings of the 2nd Language & Technology Conference*, pages 450–454, Poznań, Poland, April. Wydawnictwo Poznańskie Sp. z o.o.

Fazly, Afsaneh and Suzanne Stevenson. 2007. Distinguishing Subtypes of Multiword Expressions Using Linguistically-Motivated Statistical Measures. In *Proceedings of the Workshop on A Broader Perspective on Multiword Expressions*, pages 9–16, Prague, Czech Republic, June. Association for Computational Linguistics.

Grégoire, Nicole. 2007. Design and Implementation of a Lexicon of Dutch Multiword Expressions. In *Proceedings of the Workshop on A Broader Perspective on Multiword Expressions*, pages 17–24, Prague, Czech Republic, June. Association for Computational Linguistics.

Kaalep, Heiki-Jaan and Kadri Muischnek. 2006. Multi-Word Verbs in a Flective Language: The Case of Estonian. In *Proceedings of the EACL Workshop on Multi-Word Expressions in a Multilingual Contexts*, pages 57–64, Trento, Italy, April. Association for Computational Linguistics.

Kaalep, Heiki-Jaan and Kadri Muischnek. 2008. Multi-Word Verbs of Estonian: a Database and a Corpus. In *Proceedings of the LREC Workshop Towards a Shared Task for Multiword Expressions (MWE 2008)*, pages 23–26, Marrakech, Morocco, June.

Krenn, Brigitte. 2008. Description of Evaluation Resource – German PP-verb data. In *Proceedings of the LREC Workshop Towards a Shared Task for Multiword Expressions (MWE 2008)*, pages 7–10, Marrakech, Morocco, June.

Langer, Stefan. 2004. A Linguistic Test Battery for Support Verb Constructions. *Lingvisticae Investigationes*, 27(2):171–184.

Mel'čuk, Igor and Aleksander Žolkovskij. 1984. *Explanatory Combinatorial Dictionary of Modern Russian*. Wiener Slawistischer Almanach, Vienna, Austria.

Mel'čuk, Igor, André Clas, and Alain Polguère. 1995. *Introduction à lexicologie explicative et combinatoire*. Duculot, Louvain-la-Neuve, France.

Mel'čuk, Igor, et al. 1984–1999. *Dictionnaire explicatif et combinatoire du français contemporain: Recherches lexico-sémantiques I–IV.* Presses de l'Université de Montréal, Montreal, Canada.

Meyers, Adam, Ruth Reeves, and Catherine Macleod. 2004a. NP-External Arguments: A Study of Argument Sharing in English. In Tanaka, Takaaki, Aline Villavicencio, Francis Bond, and Anna Korhonen, editors, *Second ACL Workshop on Multiword Expressions: Integrating Processing*, pages 96–103, Barcelona, Spain, July. Association for Computational Linguistics.

Meyers, Adam, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004b. The NomBank Project: An Interim Report. In Meyers, Adam, editor, *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*, pages 24–31, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Oravecz, Csaba, Károly Varasdi, and Viktor Nagy. 2004. Többszavas kifejezések számítógépes kezelése [The treatment of multiword expressions in computational linguistics]. In Alexin, Zoltán and Dóra Csendes, editors, *MSzNy 2004 – II. Magyar Számítógépes Nyelvészeti Konferencia*, pages 141–154, Szeged, Hungary, December. University of Szeged.

Sag, Ivan A., Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2001. Multiword Expressions: A Pain in the Neck for NLP. In *Proceedings of the 3rd International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2002*, pages 1–15, Mexico City, Mexico.

Siepmann, Dirk. 2005. Collocation, colligation and encoding dictionaries. Part I: Lexicological Aspects. *International Journal of Lexicography*, 18(4):409–444.

Siepmann, Dirk. 2006. Collocation, colligation and encoding dictionaries. Part II: Lexicographical Aspects. *International Journal of Lexicography*, 19(1):1–39.

Stevenson, Suzanne, Afsaneh Fazly, and Ryan North. 2004. Statistical Measures of the Semi-Productivity of Light Verb Constructions. In Tanaka, Takaaki, Aline Villavicencio, Francis Bond, and Anna Korhonen, editors, *Second ACL Workshop on Multiword Expressions: Integrating Processing*, pages 1–8, Barcelona, Spain, July. Association for Computational Linguistics.

Tan, Yee Fan, Min-Yen Kan, and Hang Cui. 2006. Extending corpus-based identification of light verb constructions using a supervised learning framework. In *Proceedings of the EACL Workshop on Multi-Word Expressions in a Multilingual Contexts*, pages 49–56, Trento, Italy, April. Association for Computational Linguistics.

Tóth, Krisztina, Richárd Farkas, and András Kocsor. 2008. Hybrid algorithm for sentence alignment of Hungarian-English parallel corpora. *Acta Cybernetica*, 18(3):463–478.

Trón, Viktor, György Gyepesi, Péter Halácsy, András Kornai, László Németh, and Dániel Varga. 2005. hunmorph: Open Source Word Analysis. In *Proceedings of the ACL Workshop on Software*, pages 77–85, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Váradi, Tamás. 2006. Multiword Units in an MT Lexicon. In *Proceedings of the EACL Workshop on Multi-Word Expressions in a Multilingual Contexts*, pages 73–78, Trento, Italy, April. Association for Computational Linguistics.

Vincze, Veronika, Dóra Szauter, Attila Almási, György Móra, Zoltán Alexin, and János Csirik. 2010. Hungarian Dependency Treebank. In Calzolari, Nicoletta, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odjik, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May. European Language Resources Association (ELRA).

Vincze, Veronika. 2008. A puszta köznév + ige komplexumok státusáról [On the status of bare common noun + verb constructions]. In Sinkovics, Balázs, editor, *LingDok 7. Nyelvész-doktoranduszok dolgozatai*, pages 265–283, Szeged, Hungary. University of Szeged.

Vincze, Veronika. 2009. Főnév + ige szerkezetek a szótárban [Noun + verb constructions in the dictionary]. In Váradi, Tamás, editor, *III. Alkalmazott Nyelvészeti Doktorandusz Konferencia*, pages 180–188, Budapest. MTA Nyelvtudományi Intézet.

Zarrieß, Sina and Jonas Kuhn. 2009. Exploiting Translational Correspondences for Pattern-Independent MWE Identification. In *Proceedings of the Workshop on Multiword Expressions: Identification, Interpretation, Disambiguation and Applications*, pages 23–30, Singapore, August. Association for Computational Linguistics.

Syntax Based Reordering with Automatically Derived Rules for Improved Statistical Machine Translation

Karthik Visweswariah
IBM Research
v-karthik@in.ibm.com

Jiri Navratil
IBM Research
jiri@us.ibm.com

Jeffrey Sorensen
Google, Inc.
sorenj@google.com

Vijil Chenthamarakshan
IBM Research
vijil.e.c@in.ibm.com

Nanda Kambhatla
IBM Research
kambhatla@in.ibm.com

Abstract

Syntax based reordering has been shown to be an effective way of handling word order differences between source and target languages in Statistical Machine Translation (SMT) systems. We present a simple, automatic method to learn rules that reorder source sentences to more closely match the target language word order using only a source side parse tree and automatically generated alignments. The resulting rules are applied to source language inputs as a pre-processing step and demonstrate significant improvements in SMT systems across a variety of languages pairs including English to Hindi, English to Spanish and English to French as measured on a variety of internal test sets as well as a public test set.

1 Introduction

Different languages arrange words in different orders, whether due to grammatical constraints or other conventions. Dealing with these word order permutations is one of the fundamental challenges of machine translation. Given an exceptionally large training corpus, a phrase-based system can learn these reordering on a case by case basis. But, if our systems are to generalize to phrases not seen in the training data, they must explicitly capture and model these reorderings. However, permutations are difficult to model and impractical to search.

Presently, approaches that handle reorderings typically model word and phrase movements via a *distortion model* and rely on the target language model to produce words in the right order. Early distortion models simply penalized longer jumps more than shorter jumps (Koehn et al., 2003) independent of the source or target phrases in question. Other models (Tillman, 2004), (Al-Onaizan and Papineni, 2006) generalize this to include lexical dependencies on the source.

Another approach is to incorporate features, based on the target syntax, during modeling and decoding, and this is shown to be effective for various language pairs (Yamada and Knight, 2001), (Zollmann and Venugopal, 2006). Hierarchical phrase-based decoding (Chiang, 2005) also allows for long range reordering without explicitly modeling syntax. While these approaches have been shown to improve machine translation performance (Zollmann et al., 2008) they usually combine chart parsing with the decoding process, and are significantly more computationally intensive than phrase-based systems.

A third approach, one that has proved to be useful for phrase-based SMT systems, is to reorder each source-side sentence using a set of rules applied to a parse tree of the source sentence. The goal of these rules is to make the word order of the source sentence more similar to the expected target sentence word order. With this approach, the reordering rules are applied before training and testing with an SMT system. The efficacy of these methods has been shown on various language pairs including: French to English (Xia and McCord, 2004), German to English (Collins et al., 2005), English to

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1119–1127,
Beijing, August 2010

Chinese, (Wang et al., 2007) and Hindi to English (Ramanathan et al., 2008).

In this paper, we propose a simple model for reordering conditioned on the source side parse tree. The model is learned using a parallel corpus of source-target sentence pairs, machine generated word alignments, and source side parses. We apply the reordering model to both training and test data, for four different language pairs: English $\rightarrow$ Spanish, English $\rightarrow$ French, English $\rightarrow$ Hindi, and English $\rightarrow$ German. We show improvements in machine translation performance for all of the language pairs we consider except for English $\rightarrow$ German. We use this negative result to propose extensions to our reordering model. We note that the syntax based reordering we propose can be combined with other approaches to handling reordering and does not have to be followed by an assumption of monotonicity. In fact, our phrase-based model, trained upon reordered data, retains its reordering models and search, but we expect that these facilities are employed much more sparingly with reordered inputs.

2 Related work

There is a significant quantity of work in syntax based reordering employed to improve machine translation systems. We summarize our contributions to be:

- Learning the reordering rules based on training data (without relying on linguistic knowledge of the language pair)
- Requiring only source side parse trees
- Experimental results showing the efficacy for multiple language pairs
- Using a lexicalized distortion model for our baseline decoder

There have been several studies that have demonstrated improvements with syntax based reordering based upon hand-written rules. There have also been studies investigating the sources of these improvements (Zwarts and Dras, 2007). Hand-written rules depend upon expert knowledge of the linguistic properties of the particular language pair. Initial efforts (Niessen and Ney, 2001) were made at improving German-English translation by handling two phenomena: question inversion and detachable verb prefixes in German. In (Collins et al., 2005), (Wang et al., 2007), (Ramanathan et al., 2008), (Badr et al., 2009) rules are developed for translation from German to English, Chinese to English, English to Hindi, and English to Arabic respectively. (Xu et al., 2009) develop reordering rules based upon a linguistic analysis of English and Korean sentences and then apply those rules to translation from English into Korean and four other languages: Japanese, Hindi, Urdu and Turkish. Unlike this body of work, we automatically learn the rules from the training data and show efficacy on multiple language pairs.

There have been some studies that try to learn rules from the data. (Habash, 2007) learns reordering rules based on a dependency parse and they report a negative result for Arabic to English translation. (Zhang et al., 2007) learn reordering rules on chunks and part of speech tags, but the rules they learn are not hierarchical and would require large amounts of training data to learn rules for long sentences. Additionally, we only keep a single best reordering (instead of a lattice with possible reorderings) which makes the decoding significantly more efficient. (Xia and McCord, 2004) uses source and target side parse trees to automatically learn rules to reorder French sentences to match English order. The requirement to have both source and target side parse trees makes this method inapplicable to any language that does not have adequate tree bank resources. In addition, this work reports results using monotone decoding, since their experiments using non-monotone decoding without a distortion model were actually worse.

3 Reordering issues in specific languages

In this section we discuss the reordering issues typical of translating between English and Hindi, French, Spanish and German which are the four language pairs we experiment on in this paper.

3.1 Spanish and French

Typical word ordering patterns common to these two European languages relate to noun phrases including groups of nouns and adjectives. In con-

trast to English, French and Spanish adjectives and adjunct nouns follow the main noun, i.e. we typically observe a reversal of word order in noun phrases, e.g., *"A beautiful red car"* translates into French as *"Une voiture rouge beau"*, and as *"Un coche rojo bonito"* into Spanish. Phrase-based MT systems are capable of capturing these patterns provided they occur with sufficient frequency for each example in the training data. For rare noun phrases, however, the MT may produce erroneous word order that can lead to serious distortions in the meaning. Particularly difficult are nominal phrases from specialized domains that involve challenging terminology, for example: "*group reference attribute*" and "*validation checking code*". In both instances, the baseline MT system generated translations with an incorrect word order and, consequently, possibly a different meaning. We will return to these two examples in Section 5.1 to compare the output of a MT system with and without reordering.

3.2 German

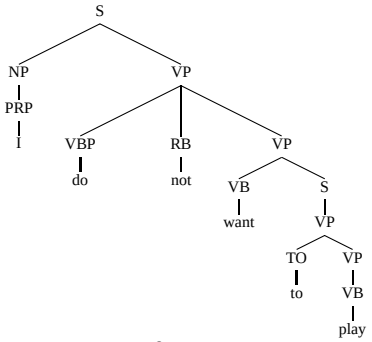
Unlike French and Spanish, German poses a considerably different challenge with respect to word ordering. The most frequent reordering in German relates to verbs, particularly verb groups consisting of auxiliary and main verbs, as well as verbs in relative clauses. Moreover, reordering patterns between German and English tend to span large portions of the sentence. We included German in our investigations to determine whether our automated rule extraction procedure can capture such long distance patterns.

3.3 Hindi

Hindi word order is significantly different than English word order; the typical order followed is Subject Object Verb (although Object Subject Verb order can be used if nouns are followed by appropriate case markers). This is in contrast to English which has a Subject Verb Object order. This can result in words that are close in English moving arbitrarily far apart in Hindi depending on the length of the noun phrase representing the object and the length of the verb phrase. These long range reorderings are generally hard for a phrase based system to capture. Another way Hindi and English differ is that prepositions in English become postpositions in Hindi and appear after the noun phrase. Again, this reordering can lead to long distance movements of words. We include Hindi in our investigation since it has significantly different structure as compared to English.

4 Learning reordering rules

In this section we describe how we learn rules that transform source parse trees so the leaf word order is more like the target language. We restrict ourselves to reorderings that can be obtained by permuting child nodes at various interior nodes in a parse tree. With many reordering phenomena discussed in Section 3 this is a fairly strong assumption about pairs of languages, and there are examples in English→Hindi where such an assumption will not allow us to generate the right reordering. As an example consider the English sentence "*I do not want to play*". The sentence has a parse:



The correct word order of the translation in Hindi is "*I to play not want*" In this case, the word *not* breaks up the verb phrase *want to play* and hence the right Hindi word order cannot be obtained by the reordering allowed by our model. We found such examples to be rare in English→Hindi, and we impose this restriction for the simplicity of the model. Experimental results on several languages show benefits of reordering in spite of this simplifying assumption.

Consider a source sentence s and its corresponding constituency parse tree S[1]. We set up the problem in a probabilistic framework, i.e. we would like to build a probabilistic model $P(T|S)$ that assigns probabilities to trees such that the

[1] In this paper we work with constituency parse trees. Initial experiments, applying similar techniques to dependency parse trees did not yield improvements.

word order in trees T which are assigned higher probability match the order of words in the target language. A parse tree, S is a set of nodes. Interior nodes have an ordered list of children. Leaf nodes in the tree are the words in the sentence s, and interior nodes are labeled by the linguistic constituent that they represent. Each word has a parent node (with only one child) labeled by the part-of-speech tag of the word.

Our model assigns non-zero probabilities to trees that can be obtained by permuting the child nodes at various interior nodes of the tree S. We assume that children of a node are ordered independently of all other nodes in the tree. Thus

$$P(T|S) = \prod_{n \in I(S)} P(\pi(c_n)|S, n, c_n),$$

where $I(S)$ is the set of interior nodes in the tree S, c_n is the list of children of node n and π is a permutation. We further assume that the reordering at a particular node is dependent only on the labels of its children:

$$P(T|S) = \prod_{n \in I(S)} P(\pi(c_n)|c_n).$$

We parameterize our model using a log-linear model:

$$P(\pi(c_n)|c_n) = \frac{1}{Z(c_n)} \exp(\lambda^T f(\pi, c_n)). \quad (1)$$

We choose the simplest possible set of feature functions: for each observed sequence of non-terminals we have one boolean feature per permutation of the sequence of non-terminals, with the feature firing iff that particular sequence is observed. Assuming, we have a training corpus C of (T, S) tree pairs, we could optimize the parameters of our model to maximize : $\prod_{S \in C} P(T|S)$. With the simple choice of feature functions described above, this amounts to:

$$P(\pi(c_n)|c_n) = \frac{\text{count}(\pi(c_n))}{\text{count}(c_n)},$$

where $\text{count}(c_n)$ is the number of times the sequences of nodes c_n is observed in the training data and $\text{count}(\pi(c_n))$ is the number of times that c_n in S is permuted to $\pi(c_n)$ in T. In Section 6, we show considering more general feature functions and relaxing some of the independence might yield improvements on certain language pairs.

For each source sentence s with parse S we find the tree T that makes the given alignment for that sentence pair most monotone. For each node n in the source tree S let D_n be the set of words that are descendants of n. Let us denote by $tpos(n)$ the average position of words in the target sentence that are aligned to words in D_n. Then

$$tpos(n) = \frac{1}{|D_n|} \sum_{w \in D_n} a(w),$$

where $a(w)$ is the index of the word on the target side that w is aligned with. If a word w is not aligned to any target word, we leave it out from the mean position calculation above. If a word w is aligned to many words we let $a(w)$ be the mean position of the words that w is aligned to. For each node n in the tree we transform the tree by sorting the list of children of n according to $tpos$. The pairs of parse trees that we obtain (S, T) in this manner form our training corpus to estimate our parameters.

In using our model, we once again go for the simplest choice, we simply reorder the source side sentences by choosing $\arg\max_T P(T|S)$ both in training and in testing; this amounts to reordering each interior node based on the most frequent reordering of the constituents seen in training. To reduce the effect of noise in training alignments we apply the reordering, only if we have seen the constituent sequence often enough in our training data (a *count threshold* parameter) and if the most frequent reordering is sufficiently more frequent than the next most frequent reordering (a *significance threshold*).

5 Experiments

5.1 Results for French, Spanish, and German

In each language, the rule extraction was performed using approximately 1.2M sentence pairs aligned using a maxent aligner (Ittycheriah and Roukos, 2005) trained using a variety of domains (Europarl, computer manuals)

and a maximum entropy parser for English (Ratnaparkhi, 1999). With a significance threshold of 1.2, we obtain about 1000 rules in the eventual reordering process.

Phrase-based systems were trained for each language pair using 11M sentence pairs spanning a variety of publicly available (e.g. Europarl, UN speeches) and internal corpora (IT technical and news domains). The system phrase blocks were extracted based on a union of HMM and maxent alignments with corpus-selective count pruning. The lexicalized distortion model was used as described in (Al-Onaizan and Papineni, 2006) with a window width of up to 5 and a maximum number of skipped (not covered) words during decoding of 2. The distortion model assigns a probability to a particular word to be observed with a specific jump. The decoder uses a 5-gram interpolated language model spanning the various domains mentioned above. The baseline system without reordering and a system with reordering was trained and evaluated in contrastive experiments. The evaluation was performed utilizing the following (single-reference) test sets:

- *News*: 541 sentences from the news domain.
- *TechA:* 600 sentences from a computer-related technical domain, this has been used as a dev set.
- *TechB:* 1038 sentences from a similar domain as *TechA* used as a blind test.
- *Dev09:* 1026 sentences defined as the *newsdev2009b* development set of the Workshop on Statistical Machine Translation 2009 [2]. This set provides a reference measurement using a public data set. Previously published results on this set can be found, for example, in (Popovic et al., 2009).

In order to assess changes in word ordering patterns prior to and after an application of the reordering, we created histograms of word jumps in the alignments obtained in the baseline as well as in the reordered system. Given a source word s_i at index i and the target word t_j it is aligned to at index j, a jump of 1 would correspond to s_{i+1} aligning to target word t_{j+1}, while an alignment to t_{j-1} corresponds to a jump of -1, etc. A

[2]http://statmt.org/wmt09/

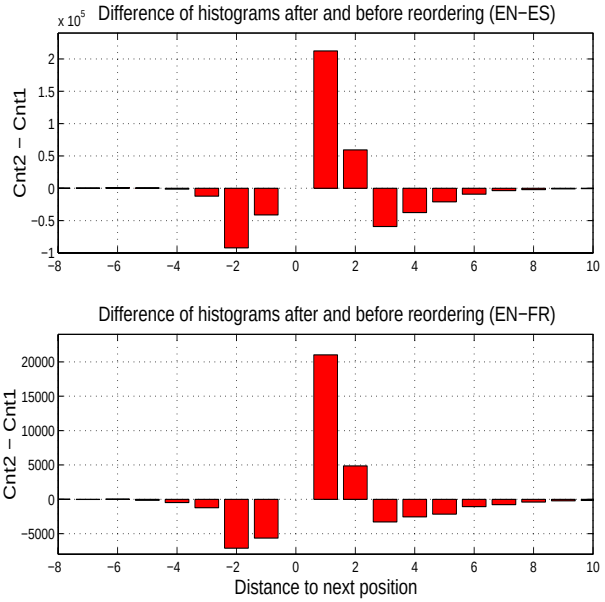


Figure 1: Difference-histogram of word order distortions for English→Spanish (upper), and English→French (lower).

histogram over the jump values gives us a summary of word order distortion. If all of the jumps were one, then there is no reordering between the two languages. To gain insight into changes introduced by our reordering we look at differences of the two histograms i.e., counts after reordering minus counts before reordering. We would hope that after reordering most of the jumps are small and concentrated around one. Figure 1 shows such *difference*-histograms for the language pairs English→Spanish and English→French, respectively, on a sample of about 15k sentence pairs held out of the system training data. Here, a positive difference value indicates an increased number *after* reordering. In both cases a consistent trend toward monotonicity is observed, i.e more jumps of size one and two, and fewer large jumps. This confirms the intended reordering effect and indicates that the reordering rules extracted generalize well.

Table 1 shows the resulting uncased BLEU scores for English-Spanish and English-French.

In both cases the reordering has a consistent positive effect on the BLEU scores across test sets. In examining the sources of improvement, we noticed that word order in several noun phrases that

	System	News	TechA	TechB	Dev09
Spanish	Baseline	0.3849	0.3371	0.3483	0.2244
	Reordered	0.4031	0.3582	0.3605	0.2320
French	Baseline	0.5140	0.2971	0.3035	0.2014
	Reordered	0.5242	0.3152	0.3154	0.2092
German	Baseline	0.2580	0.1582	0.1697	0.1281
	Reordered	0.2544	0.1606	0.1682	0.1271
Hindi	Baseline	20.0			
	Reordered	21.7			

Table 1: Uncased BLEU scores for phrase-based machine translation.

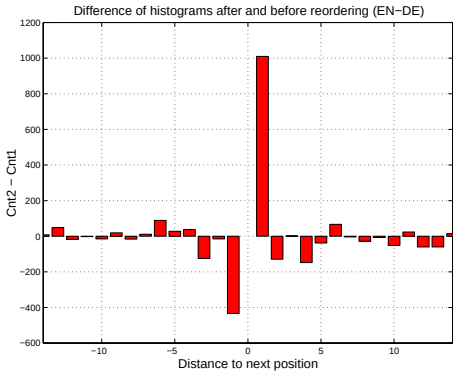


Figure 2: Difference-histogram of word order distortions for English→German.

were not common in the training data were fixed by use of the reordering rules.

Table 1 shows the BLEU scores for the English→German language pair, for which a mixed result is observed. The difference-histogram for English→German, shown in Figure 2, differs from those of the other languages with several increases in jumps of large magnitude, indicating failure of the extracted rules to generalize.

The failure of our simple method to gain consistent improvements comparable to Spanish and French, along with our preliminary finding that a relatively few manually crafted reordering rules (we describe these in Section 6.4) tend to outperform our method, leads us to believe that a more refined approach is needed in this case and will be subject of further discussion below.

5.2 Results for Hindi

Our Hindi-English experiments were run with an internal parallel corpus of roughly 250k sentence pairs (5.5M words) consisting of various domains (including news). To learn reordering rules we used HMM alignments and a maxent parser (Ratnaparkhi, 1999), with a count threshold of 100, and a significance threshold of 1.7 (these settings gave us roughly 200 rules). We also experimented with other values of these thresholds and found that the performance of our systems were not very sensitive to these thresholds. We trained Direct Translation Model 2 (DTM) systems (Ittycheriah and Roukos, 2007) with and without source reordering and evaluated on a test set of 357 sentences from the News domain. We note that the DTM baseline includes features (functions of target words and jump size) that allow it to model lexicalized reordering phenomena. The reordering window size was set to +/- 8 words for the baseline and system with reordered inputs. Table 1 shows the uncased BLEU scores for English-Hindi, showing a gain from using the reordering rules. For the reordered case, the HMM alignments are rederived, but the accuracy of these were no better than those of the unreordered input and experiments showed that the gains in performance were not due to the effect on the alignments.

Figure 3 shows *difference*-histograms for the language pair English→Hindi, on a sample of about 10k sentence pairs held out of the system training data. The histogram indicates that our reordering rules generalize and that the reordered English is far more monotonic with respect to the Hindi.

6 Analysis of errors and future directions

In this section, we analyze some of the sources of errors in reordering rules learned via our model, to better understand directions for further improvement.

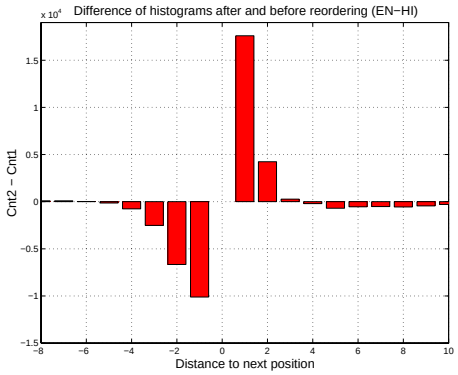


Figure 3: Difference-histogram of word order distortions for English→Hindi.

6.1 Model weakness

In our initial experiments, we noticed that for the most frequent reordering rules in English→Hindi (e.g that IN NP or NP PP flips in Hindi) the probability of a reordering was roughly 65%. This was concerning since it meant that on 35% of the data we would be making wrong reordering decisions by choosing the most likely reordering. To get a better feel for whether we needed a stronger model (e.g by lexicalization or by looking at larger context in the tree rather than just the children), we analyzed some of the cases in our training data where (IN,NP), (NP, PP) pairs were left unaltered in Hindi. In doing that analysis, we noticed examples involving negatives that our model does not currently handle. The first issue was mentioned in Section 4, where the assumption that we can achieve the right word order by reordering constituent phrases, is incorrect. The second issue is illustrated by the following sentences: *I have some/no books*, which have similar parse structures, the only difference being the determiner *some* vs the determiner *no*. In Hindi, the order of the fragments *some books* and the fragment *no books* are different (in the first case the words stay in order, in the second the flip). Handling this example would need our model to be lexicalized. These issue of negatives requiring special handling also came up in our analysis of German (Section 6.4). Other than the negatives (which require a lexicalized model), the major reason for the lack of sharpness of the reordering rule probability was alignment errors and parser issues. We look at these topics next.

Aligner	Number of Sentences	fMeasure	BLEU score
HMM	250k	62.4	21.7
MaxEnt	250k	76.6	21.4
Manual	5k	-	21.3

Table 2: Using different alignments

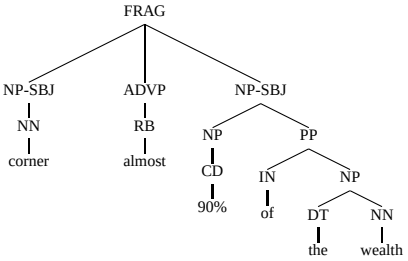
6.2 Alignment accuracy

Since we rely on automatically generated alignments to learn the rules, low accuracy of the alignments could impact the quality of the rules learned. This is especially a concern for English→Hindi since the quality of HMM alignments are fairly low. To quantify this effect, we learn reordering rules using three sets of alignments: HMM alignments, alignments from a supervised MaxEnt aligner (Ittycheriah and Roukos, 2005), and hand alignments. Table 2 summarizes our results using aligners with differing alignment qualities for our English→Hindi task and shows that quality of alignments in learning the rules is not the driving factor in affecting rule quality.

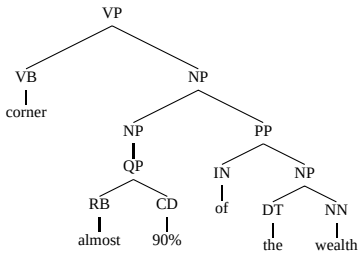
6.3 Parser accuracy

Accuracy of the parser in the source language is a key requirement for our reordering method, because we choose the single best reordering based on the most likely parse of the source sentence. This would especially be an issue in translating *from* languages other than English, where the parser would not be of quality comparable to the English parser.

In examining some of the errors in reordering we did observe a fair fraction attributable to issues in parsing, as seen in the example sentence: *The rich of this country , corner almost 90% of the wealth .*
The second half of the sentence is parsed by the Berkeley parser (Petrov et al., 2006) as:



and by IBM's maximum entropy parser parser (Ratnaparkhi, 1999) as:



With the first parse, we get the right Hindi order for the second part of the sentence which is: *the wealth of almost 90% corner* . To investigate the effect of choice of parser we compared using the Berkeley parser and the IBM parser for reordering, and we found the BLEU score essentially unchanged: 21.6 for the Berkeley parser and 21.7 for the IBM parser. A potential source of improvements might be to use alternative parses (via different parsers or n-best parses) to generate n-best reorderings both in training and at test.

6.4 Remarks on German reordering

Despite a common heritage, German word order is distinct from English, particularly regarding verb placement. This difference can be dramatic, if an auxiliary (e.g. modal) verb is used in conjunction with a full verb, or the sentence contains a subordinate clause. In addition to our experiments with automatically learned rules, a small set of handcrafted reordering rules was created and evaluated. Our preliminary results indicate that the latter rules tend to outperform the automatically derived ones by 0.5-1.0 BLEU points on average. These rules are summarized as follows:

1. In a VP immediately following an NP, move the negation particle to main verb.
2. Move a verb group away from a modal verb; to the end the of a VP. Negation also moves along with verb.
3. Move verb group to end of an embedded/relative clause.
4. In a VP following a subject, move negation to the end of VP (handling residual cases)

The above hand written rules show several weaknesses of our automatically learned rules for reordering. Since our model is not lexicalized, negations are not handled properly as they are tagged RB (along with other adverbs). Another limitation apparent from the first rule above (the movement of verbs in a verb phrase depends on the previous phrase being a noun phrase) is that the automatic reordering rule for a node's children depends only on the children of that node and not a larger context. For instance, a full verb following a modal verb is typically parsed as a VP child node of the modal VP node, hence the automatic rule, as currently considered, will not take the modal verb (being a sibling of the full-verb VP node) into account. We are currently investigating extensions of the automatic rule extraction alorithm to address these shortcomings.

6.5 Future directions

Based on our analysis of the errors and on the hand designed German rules we would like to extend our model with more general feature functions in Equation 1 by allowing features: that are dependent on the constituent words (or headwords), that examine a large context than just a nodes children (see the first German rule above) and that fire for all permutations when the constituent X is moved to the end (or start). This would allow us to generalize more easily to learn rules of the type "move X to the end of the phrase". Another direction that we feel should be explored, is the use of multiple parses to obtain multiple reorderings and combine these at a later stage.

7 Conclusions

In this paper we presented a simple method to automatically derive rules for reordering source sentences to make it look more like target language sentences. Experiments (on internal and public test sets) indicate performance gains for English→French, English→Spanish, and English→Hindi. For English→German we did not see improvements with automatically learned rules while a few hand designed rules did give improvements, which motivated a few directions to explore.

References

[Al-Onaizan and Papineni2006] Al-Onaizan, Yaser and Kishore Papineni. 2006. Distortion models for statistical machine translation. In *Proceedings of ACL*.

[Badr et al.2009] Badr, Ibrahim, Rabih Zbib, and James Glass. 2009. Syntactic phrase reordering for english-to-arabic statistical machine translation. In *Proceedings of EACL*.

[Chiang2005] Chiang, David. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL*.

[Collins et al.2005] Collins, Michael, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of ACL*.

[Habash2007] Habash, Nizar. 2007. Syntactic preprocessing for statistical machine translation. In *MT Summit*.

[Ittycheriah and Roukos2005] Ittycheriah, Abraham and Salim Roukos. 2005. A maximum entropy word aligner for arabic-english machine translation. In *Proceedings of HLT/EMNLP*.

[Ittycheriah and Roukos2007] Ittycheriah, Abraham and Salim Roukos. 2007. Direct translation model 2. In *Proceedings of HLT-NAACL*, pages 57–64.

[Koehn et al.2003] Koehn, Philipp, Franz Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL*.

[Niessen and Ney2001] Niessen, Sonja and Hermann Ney. 2001. Morpho-syntactic analysis for reordering in statistical machine translation. In *Proc. MT Summit VIII*.

[Petrov et al.2006] Petrov, Slav, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *COLING-ACL*.

[Popovic et al.2009] Popovic, Maja, David Vilar, Daniel Stein, Evgeny Matusov, and Hermann Ney. 2009. The RWTH machine translation system for WMT 2009. In *Proceedings of WMT 2009*.

[Ramanathan et al.2008] Ramanathan, A., P. Bhattacharyya, J. Hegde, R. M. Shah, and M. Sasikumar. 2008. Simple syntactic and morphological processing can help english-hindi statistical machine translation. In *Proceedings of International Joint Conference on Natural Language Processing*.

[Ratnaparkhi1999] Ratnaparkhi, Adwait. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34(1-3).

[Tillman2004] Tillman, Christoph. 2004. A unigram orientation model for statistical machine translation. In *Proceedings of HLT-NAACL*.

[Wang et al.2007] Wang, Chao, Michael Collins, and Philipp Koehn. 2007. Chinese syntactic reordering for statistical machine translation. In *Proceedings of EMNLP-CoNLL*.

[Xia and McCord2004] Xia, Fei and Michael McCord. 2004. Improving a statistical mt system with automatically learned rewrite patterns. In *Proceedings of Coling*.

[Xu et al.2009] Xu, Peng, Jaeho Kang, Michael Ringgaard, and Franz Och. 2009. Using a dependency parser to improve SMT for Subject-Object-Verb languages. In *Proceedings of NAACL-HLT*.

[Yamada and Knight2001] Yamada, Kenji and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of ACL*.

[Zhang et al.2007] Zhang, Yuqi, Richard Zens, and Hermann Ney. 2007. Chunk-level reordering of source language sentences with automatically learned rules for statistical machine translation. In *NAACL-HLT AMTA Workshop on Syntax and Structure in Statistical Translation*.

[Zollmann and Venugopal2006] Zollmann, Andreas and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings on the Workshop on Statistical Machine Translation*.

[Zollmann et al.2008] Zollmann, Andreas, Ashish Venugopal, Franz Och, and Jay Ponte. 2008. A systematic comparison of phrase-based, hierarchical and syntax-augmented statistical MT. In *Proceedings of COLING*.

[Zwarts and Dras2007] Zwarts, Simon and Mark Dras. 2007. Syntax-based word reordering in phrase-based statistical machine translation: why does it work? In *Proc. MT Summit*.

Efficient Statement Identification for Automatic Market Forecasting

Henning Wachsmuth
Universität Paderborn
Software Quality Lab
hwachsmuth@slab.upb.de

Peter Prettenhofer and **Benno Stein**
Bauhaus-Universität Weimar
Web Technology & Information Systems
benno.stein@uni-weimar.de

Abstract

Strategic business decision making involves the analysis of market forecasts. Today, the identification and aggregation of relevant market statements is done by human experts, often by analyzing documents from the World Wide Web. We present an efficient information extraction chain to automate this complex natural language processing task and show results for the identification part. Based on time and money extraction, we identify sentences that represent statements on revenue using support vector classification. We provide a corpus with German online news articles, in which more than 2,000 such sentences are annotated by domain experts from the industry. On the test data, our statement identification algorithm achieves an overall precision and recall of 0.86 and 0.87 respectively.

1 Introduction

Touch screen market to hit $9B by 2015. 50 suppliers provide multi-touch screens, and that number is likely to rise.[1]

Strategic business decision making is a highly complex process that requires experience as well as an overall view of economics, politics, and technological developments. Clearly, for the time being this process cannot be done by a computer at the level of a human expert. However, important tasks may be automated such as market forecasting, which relies on identifying and aggregating relevant information from the World Wide Web (Berekoven et. al., 2001). An analyst who interprets the respective data can get a reasonable idea about the future market volume, for example. The problem is that a manually conducted Web search is time-consuming and usually far from being exhaustive. With our research we seek to develop an efficient system that finds and analyzes market forecast information with retrieval, extraction and natural language processing (NLP) techniques.

We contribute to the following situation. For a given product, technology, or industry sector we identify and aggregate statements on its market development found on relevant websites. In particular, we extract time information ("by 2015") and money information ("$9B") and use support vector classification to identify sentences that represent market statements. The statements' subjects ("touch screen") are found by relating recognized named entities to the time and money information, which we then normalize and aggregate. In this paper we report on results for the statement identification. To the best of our knowledge no data for the investigation of such market analysis tasks has been made publicly available until now. We provide such a corpus with statements on revenue annotated in news articles from the Web; the corpus was created in close collaboration with our industry partner *Resolto Informatik GmbH*.

We pursue two objectives, namely, to support human experts with respect to the effectiveness and completeness of their analysis, and to establish a technological basis upon which more intricate analysis tasks can be automated. To summarize, the main contributions of this paper are:

1. We show how to decompose the identification and aggregation of forecasts into retrieval, extraction, and normalization tasks.
2. We introduce a manually annotated German corpus for computational linguistics research on market information.
3. We offer empirical evidence that classification and extraction techniques can be com-

[1] Adapted from http://industry.bnet.com.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1128–1136,
Beijing, August 2010

bined to precisely identify statements on revenue.

1.1 Related Work

Stein et. al. (2005) were among the first to consider information extraction for automatic market forecasting. Unlike us, the authors put much emphasis on retrieval aspects and applied dependency grammar parsing to identify market statements. As a consequence their approach suffers from the limitation to a small number of predefined sentence structures.

While we obtain market forecasts by extracting expert statements from the Web, related approaches derive them from past market behavior and quantitative news data. Koppel and Shtrimberg (2004) studied the effect of news on financial markets. Lavrenko et al. (2000) used time-series analysis and language models to predict stock market prices and, similarly, Lerman et al. (2008) proposed a system for forecasting public opinion based on concurrent modeling of news articles and market history. Another related field is opinion mining in the sense that it relies on the aggregation of individual statements. Glance et al. (2005) inferred marketing intelligence from opinions in online discussions. Liu et al. (2007) examined the effect of Weblogs on box office revenues and combined time-series with sentiment analysis to predict the sales performance of movies.

The mentioned approaches are intended to reflect or to predict present developments and, therefore, primarily help for *operative* decision making. In contrast, we aim at predicting long-term market developments, which are essential for *strategic* decision making.

2 The Problem

Market forecasts depend on two parameters, the *topic* of interest and the *criterion* to look at. A topic is either an organization or a market. Under a market we unite branches, products, and technologies, because the distinction between these is not clear in general (e.g., for semiconductors). In contrast, we define a criterion to be a metric attribute that can be measured over time. Here we are interested in financial criteria such as revenue, profit, and the like. The ambitious overall task that we want to solve is as follows:

Task description: Given a topic τ and a financial criterion χ, find information for τ on the development of χ. Aggregate the found values on χ with respect to time.

We omit the limitation to forecasts because we could miss useful information otherwise:

(1) *In 2008*, the Egyptian automobile industry achieved *US$ 9.96bn* in sales.

(2) Egypt's automotive sales will rise *by 97% from 2008 to 2013*.

Both sentences have the same topic. In Particular, the 2008 amount of money from example (1) can be aggregated with the forecast in (2) to infer the predicted amount in 2013.

As in these examples, market information can often only be found in running text; the major source for this is the Web. Thus, we seek to find web pages with sentences that represent *statements on a financial criterion* χ and to make these statements processable. Conceptually, such a statement is a 5-tuple $\mathcal{S}_\chi = (S, g, T, M, t_d)$, where S is the topical subject, which may have a geographic scope g, T is a period of time, M consists of a growth rate and/or an amount of money to be achieved during T with respect to χ, and t_d is the statement time, i.e., the point in time when the statement was made.

3 Approach

Our goal is to find and aggregate statements on a criterion χ for a topic τ. In close collaboration with two companies from the semantic technology field, we identified eight high-level subtasks in the overall process as explained in the following. An overview is given in Table 1.

3.1 Find Candidate Documents

To find web pages that are likely to contain statements on χ and τ, we propose to perform a meta-search by starting from a set of characteristic terms of the domain and then using query expansion techniques such as local context analysis (Xu and Croft, 2000). As Stein et. al. (2005) describe,

Subtask	Applied technologies
1 Find candidate documents	meta-search, query expansion, genre analysis
2 Preprocess content	content extraction, sentence splitting, tokenization, POS tagging and chunking
3 Extract entities	time and money extraction, named entity recognition of organizations and markets
4 Identify statements	statistical classification based on lexical and distance features
5 Determine statement type	relation extraction based on dependency parse trees, matching of word lists
6 Fill statement templates	template filling, anaphora resolution, matching of word lists
7 Normalize values	time and money normalization, coreference resolution
8 Aggregate information	chronological merging and averaging, inference from subtopic to topic

Table 1: Subtasks of the identification and aggregation of market statements for a specified topic. Experiments in this paper cover the subtasks written in black.

a genre analysis, which classifies a document with respect to its form, style, and targeted audience, may be deployed afterwards to further improve the quality of the result list efficiently. In this way, we only maintain candidate documents that look promising on the surface.

3.2 Preprocess Content

Preprocessing is needed for accurate access to the document text. Our overall task incorporates relating information from different document areas, so mixing up a web page's main frame and sidebars should be avoided. We choose Document Slope Curve (DSC) for content detection, which looks for plateaus in the HTML tag distribution. Gottron (2007) has offered evidence that DSC is currently the best algorithm in terms of precision. Afterwards, the sentences are split with rules that consider the specific characteristics of reports, press releases and the like, such as headlines between short paragraphs. In succeeding subtasks, tokens as well as their Part-of-Speech and chunk tags are also used, but we see no point in not relying on standard algorithms here.

3.3 Extract Entities

The key to identify a statement $\mathcal{S}_\chi$ on a financial criterion χ is the extraction of temporal and monetary entities. Recent works report that statistical approaches to this task can compete with hand-crafted rules (Ahn et. al., 2005; Cramer et. al., 2007). In the financial domain, however, the focus is only on dates and periods as time information, along with currency numbers, currency terms, or fractions as money information. We found that with regular expressions, which represent the complex but finite structures of such phrases, we can achieve nearly perfect recall in recognition (see Section 5).

We apply named entity recognition (NER) of organizations and markets in this stage, too, so we can relate statements to the appropriate subjects, later on. Note that market names do not follow a unique naming scheme, but we observed that they often involve similar phrase patterns that can be exploited as features. NER is usually done by sequence labeling, and we use heuristic beam search due to our effort to design a highly efficient overall system. Ratinov and Roth (2009) have shown for the CoNLL-2003 shared task that Greedy decoding (i.e., beam search of width 1) is competitive to the widely used Viterbi algorithm while being over 100 times faster at the same time.

3.4 Identify Statements

Based on time and money information, sentences that represent a statement $\mathcal{S}_\chi$ can be identified. Such a sentence gives us valuable hints on which temporal and monetary entity stick together and how to interpret them in relation. Additionally, it serves as evidence for the statement's correctness (or incorrectness). Every sentence with at least one temporal and one monetary entity is a candidate. Criteria such as revenue usually imply small core vocabularies $\mathcal{L}_{pos}$, which indicate that a sentence is on that criterion or which often appear close to it. On the contrary, there are sets of words $\mathcal{L}_{neg}$ that suggest a different criterion. For a given text collection with known statements on χ, both $\mathcal{L}_{pos}$ and $\mathcal{L}_{neg}$ can be found by computing the most discriminant terms with respect to χ. A reasonable first approach is then to filter sentences

that contain terms from $\mathcal{L}_{pos}$ and lack terms from $\mathcal{L}_{neg}$, but problems arise when terms from different vocabularies co-occur or statements on different criteria are attached to one another.

Instead, we propose a statistical learning approach. Support Vector Machines (SVMs) have been proven to yield very good performance in both general classification and sentence extraction while being immune to overfitting (Steinwart and Christmann, 2008; Hirao et. al., 2001). For our candidates, we compute lexical and distance features based on $\mathcal{L}_{pos}$, $\mathcal{L}_{neg}$, and the time and money information. Then we let an SVM use these features to distinguish between sentences with statements on χ and others. At least for online news articles, this works reasonably well as we demonstrate in Section 5. Note that classification is not used to match the right entities, but to filter the small set of sentences on χ.

3.5 Determine Statement Type

The statement type implies what information we can process. If a sentence contains more than one temporal or monetary entity, we need to relate the correct T and M to each $\mathcal{S}_\chi$, now. The type of $\mathcal{S}_\chi$ then depends on the available money information, its *trend* and the *time direction*.

We consider four types of money information. χ refers to a period of time that results in a *new amount* A of money in contrast to its *preceding amount* A_p. The difference between A and A_p may be specified as an *incremental amount* Δ_A or as a *relative growth rate* r. M can span any combination of A, A_p, Δ_A and r, and at least A and r constitute a reasonable entity on their own. Sometimes the trend of r (i.e. decreasing or increasing) cannot be derived from the given values. However, this information can mostly be obtained from a nearby indicator word (e.g. "plus" or "decreased") and, therefore, we address this problem with appropriate word lists. Once the trend is known, any two types imply the others.

Though we are predominantly interested in *forecasts*, statements also often represent a *declaration* on achieved results. This distinction is essential and can be based on time-directional indicators (e.g. "next") and the tense of leading verbs. For this, we test both feature and kernel methods on dependency parse trees, thereby determining T and M at the same time. We only parse the identified sentences, though. Hence, we avoid running into efficiency problems.

3.6 Fill Statement Templates

The remaining subtasks are ongoing work, so we only present basic concepts here.

Besides T and M, the subject S and the statement time t_d have to be determined. S may be found within the previously extracted named entities using the dependency parse tree from Section 3.5 or by anaphora resolution. Possible limitations to a geographic scope g can be recognized with word lists. In market analysis, the approximate t_d suffices, and for most news articles t_d is similar to their release date. Thus, if no date is in the parse tree, we search the extracted temporal entities for the release date, which is often mentioned at the beginning or end of the document's content. We fill one template (S, g, T, M, t_d) for each $\mathcal{S}_\chi$ where we have at least S, T, and M.

3.7 Normalize Values

Since we base the extraction on regular expressions, we can normalize most monetary entities with a predefined set of rules. Section 3.5 implies that $M^* = (A^*, r^*)$ is a reasonable normalized form where A^* is A specified in million US-$ and r^* is r as percentage with a fixed number of decimals.[2] Time normalization is more complex. Any period should be transformed to $T^* = (t_s^*, t_e^*)$ consisting of the start date t_s^* and end date t_e^*. Following Ahn et. al. (2005), we consider fully qualified, deictic and anaphoric periods. While normalization of fully qualified periods like "from Apr to Jun 1999" is straightforward, deictic (e.g. "since 2005", "next year") and anaphoric mentions (e.g. "in the reported time") require a reference time. Approaches to resolve such references rely on dates or fully qualified periods in the preceding text (Saquete et. al., 2003; Mani and Wilson, 2000).[3]

[2] Translating the currency requires exchange rates at statement time. We need access to such information or omit the translation if only one currency is relevant.

[3] References to fiscal years even involve a whole search problem if no look-up table on such data is available.

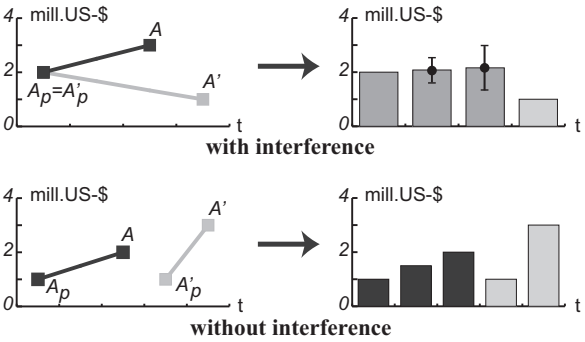


Figure 1: Example for merging monetary values.

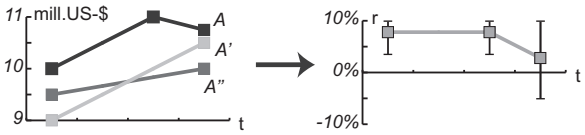


Figure 2: Example for the inference of relative information from absolute values.

If we cannot normalize M or T, we discard the corresponding statement templates. For the others, we have to resolve synonymous co-references (e.g. "Loewe AG" and "Loewe") before we can proceed to the last step.

3.8 Aggregate Information

We can aggregate the normalized values in either two or three dimensions depending on whether to separate statements with respect to t_d. Aggregation then incorporates two challenges, namely, how to merge values and how to infer information on a topic from values of a subtopic.

We say that two statements on the same topic τ and criterion χ *interfere* if the contained periods of time intersect and the according monetary values do not coincide. In case of declarations, this means that we extracted incorrect values or extracted values incorrectly. For forecasts, on the contrary, we are exactly onto such information. In both cases, an intuitive solution is to compute the average (or median) and deviations. Figure 1 graphically illustrates such merging. The subtopic challenge is based on the assumption that a meaningful number of statements on a certain subtopic of τ implies relative information on τ, as shown in Figure 2. One of the most interesting relations are organizations as subtopics of markets they produce for, because it is quite usual to search for information on a market, but only receive statements on companies. Approaches to this relation may rely e.g. on the web page co-occurrence and term frequencies of the markets and companies.

Statements	**Total**	**Forecasts**	**Declarations**
Complete corpus	2075	523 (25.2%)	1552 (74.8%)
Training set	1366	306 (22.4%)	1060 (77.6%)
Validation set	362	113 (31.2%)	249 (68.8%)
Test set	347	104 (30.0%)	243 (70.0%)

Table 2: Statements on revenue in the corpus.

Altogether, we return the aggregated values linked to the sentences in which we found them. In this way, we make the results verifiable and, thereby, compensate for possible inaccuracies.

4 Corpus

To evaluate the given and related tasks, we built a manually annotated corpus with online news articles on the revenues of organizations and markets. The compilation aims at being representative for target documents, a search engine returns to queries on revenue. The purpose of the corpus is to investigate both the structure of sentences on financial criteria and the distribution of associated information over the text.

The corpus consists of 1,128 German news articles from the years 2003 to 2009, which were taken from 29 news websites like *www.spiegel.de* or *www.capital.de*. The content of each document comes as unicode plain text with appended URL for access to the HTML source code. Annotations are given in a standard XMI file preformatted for the *Unstructured Information Management Architecture* (Ferrucci and Lally, 2004). We created a split, in which 2/3 of the documents constitute the training set and each 1/6 refers to the validation and test set. To simulate real conditions, the training documents were randomly chosen from only the seven most represented websites, while the validation and test data both cover all 29 sources. Table 2 shows some corpus statistics, which give a hint that the validation and test set differ significantly from the training set. The corpus is free for scientific use and can be downloaded at http://infexba.upb.de.

Loewe AG: Vorläufige Neun-Monats-Zahlen

Kronach, [6. November 2007]$_{REF}$ — Das Ergebnis vor Zinsen und Steuern (EBIT) des Loewe Konzerns konnte in den ersten 9 Monaten 2007 um 41% gesteigert werden. Vor diesem Hintergrund hebt die [Loewe AG]$_{ORG}$ ihre EBIT-Prognose für das laufende Geschäftsjahr auf 20 Mio. Euro an. ***Beim Umsatz strebt Konzernchef [Rainer Hecker]$_{AUTH}$ [für das Gesamtjahr]$_{TIME}$ ein höher als ursprünglich geplantes [Wachstum]$_{TREND}$ [von 10% auf ca. 380 Mio. Euro]$_{MONEY}$ an.*** *(...)*

Figure 3: An annotated document in the corpus. The text is taken from *www.boerse-online.de*, but has been modified for clarification.

4.1 Annotations

In each document, every sentence that includes a temporal entity T and a monetary entity M and that represents a *forecast* or *declaration* on the revenue of an organization or market is marked as such. T and M are annotated themselves and linked to the sentence. Accordingly, the *subject* is tagged (and linked) within the sentence boundaries if available, otherwise its last mention in the preceding text. The same holds for optional entities, namely a *reference time*, a *trend indicator* and the *author* of a statement. Altogether, 2,075 statements are tagged in this way. As in Figure 3, only information that refers to a statement on revenue (typed in bold face) is annotated. These annotations may be spread across the text.

The source documents were manually selected and prepared by our industrial partners, and two of their employees annotated the plain document text. With respect to the statement annotations, a preceding pilot study yielded substantial inter-annotator agreement, as indicated by the value $\kappa = 0.79$ of the conservative measure *Cohen's Kappa* (Carletta, 1996). Additionally, we performed a manual correction process for each annotated document to improve consistency.

5 Experiments

We now present experiments for the statement identification, which were conducted on our corpus. The goal was to evaluate whether our combined extraction and classification approach succeeds in the precise identification of sentences that comprise a statement on revenue, while keeping recall high. Only exact matches of the annotated text spans were considered to be correct identifications. Unlike in Section 3, we only worked on plain text, though.

5.1 Experimental Setup

To find candidate sentences, we implemented a sentence splitter that can handle article elements such as subheadings, URLs, or bracketed sentences. We then constructed sophisticated, but efficient regular expressions for time and money. They do not represent correct language, in general, but model the structure of temporal and monetary entities, and use word lists provided by domain experts on the lowest level.[4] For feature computation, we assumed that the closest pair of temporal and monetary entity refers to the enclosing candidate sentence.[5] Since only positive instances I_P of statements on revenue are annotated in our corpus, we declared all candidates, which have no counterpart in the annotated data, to constitute the negative class I_N, and balanced I_P and I_N by "randomly" (seed 42) removing instances from I_N.[6]

For the vocabularies $\mathcal{L}_{pos} = \{P_1, P_2\}$ we first counted the frequencies of all words in the unbalanced sets I_P and I_N. From these, we deleted named entities, numbers and adjectives. If the prefix (e.g. "Umsatz") of a word ("Umsatzplus") occurred, we only kept the prefix. We then filtered all terms that appeared in at least 1.25% of the instances in I_P and more than 3.5 times as much in I_P as in I_N. The remaining words were manually partitioned into two lists:

P_1 = {umgesetzt, Umsatz, Umsätze, setzte} (all of these are terms for revenue)

P_2 = {Billionen, meldet, Mitarbeiter, Verband} (trillions, announce, employee, association)

$\mathcal{L}_{neg} = \{N_1, N_2\}$ was built accordingly. In addition, we set up a list G_1 with genitive pronouns

[4] More details are given at http://infexba.upb.de.

[5] 55% of the candidate sentences in the training set contain more than one temporal and/or monetary entity, so this assumption may lead to errors.

[6] We both tested undersampling and oversampling techniques but saw no effective differences in the results.

and determiners. Based on $\mathcal{L}_{pos}$, $\mathcal{L}_{neg}$ and G_1, we computed the following 43 features for every candidate sentence s:

- **1-8:** Number of terms from P_1 (N_1) in s as well as in the two preceding sentences and in the following sentence.
- **9-10:** Number of terms from P_2 (N_2) in s.
- **11:** Occurrence of term from G_1 next to the monetary entity.
- **12-19:** Forward (backward) distance in tokens between the monetary (temporal) entity in s and a term from P_1 (N_1).
- **20-27:** Forward (backward) distance in number of symbols from $O_1 = \{$'.','?','!'$\}$ between the monetary (temporal) entity in s and a term from P_1 (N_1).
- **28-43:** Same as 20-27 for $O_2 = \{$':',';'$\}$ and $O_3 = \{$','$\}$, respectively.

We trained a linear SVM with cost parameter $C = 0.3$ (selected during validation) on these features using the *Weka* integration of *LibSVM* (Hall et. al., 2009; Fan et. al., 2001). Further features were evaluated, e.g. occurrences of contrapositions or comparisons, but they did not improve the classifier. Instead, we noticed that we can avoid some complex cases when we apply two rules after entity extraction:

R_1: Delete temporal and monetary entities that are directly surrounded by brackets.

R_2: Delete temporal entities that contain the word "Vorjahr" ("preceding year").

Now, we evaluated the following five statement identification algorithms:

- **Naïve:** Simply return all candidate sentences (to estimate the relative frequency of statements on revenue in the corpus).
- **Baseline:** Return all candidate sentences that contain a term from the list P_1.
- **NEG:** Use the results from Baseline. Return all sentences that lack terms from N_1.

Recall	Training	Validation	Test
Sentences	0.98	0.98	0.96
Temporal entities	0.97 (0.95)	0.97 (0.94)	0.98 (0.96)
Monetary entities	0.96 (0.96)	0.96 (0.96)	0.95 (0.94)

Table 3: Recall of sentence and entity extraction. In brackets: Recall after applying R_1 and R_2.

- **RB:** Filter candidates using R_1 and R_2. Then apply NEG.
- **SVM:** Filter candidates using R_1 and R_2. Then classify sentences with the SVM.

5.2 Results

Table 3 shows that we found at least 95% of the sentences, time and money information, which refer to a statement on revenue, in all datasets.[7] We could not measure precision for these since not all sentences and entities are annotated in the corpus, as mentioned in Section 4.

Results for the statement identification are given in Figure 4. Generally, the test values are somewhat lower than the validation values, but analog in distribution. Nearly all statements were recognized by the Naïve algorithm, but only with a precision of 0.35. In contrast, both for Baseline and NEG already around 80% of the found statements were correct. The latter paid a small gain in precision with a significant loss in recall. While RB and SVM both achieved 86% precision on the test set, SVM tends to be a little more precise as suggested by the validation results. In terms of recall, SVM clearly outperformed RB with values of 89% and 87% and was only a little worse than the Baseline. Altogether, the F_1-Measure values show that SVM was the best performing algorithm in our evaluation.

5.3 Error Analysis

To assess the influence of the sentence, time and money extraction, we compared precision and recall of the classifier on the manually annotated and the extracted data, respectively. Table 4 shows

[7] We intentionally did not search for unusual entities like "am 1. Handelstag nach dem Erntedankfest" ("the 1st trading day after Thanksgiving") in order not to develop techniques that are tailored to individual cases. Also, money amounts that lack a currency term were not recognized.

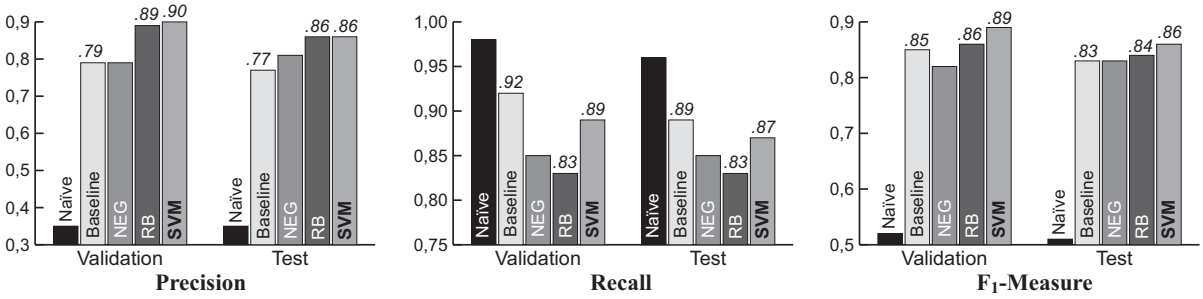


Figure 4: Precision, recall and F_1-Measure of the five evaluated statement identification algorithms. SVM is best in precision both on validation and test data and outperforms RB in recall significantly.

that only recall differs significantly. We found that false statement identifications referred to the following noteworthy error cases.

False match: Most false positives result from matchings of temporal and monetary entities that actually do not refer to the same statement.

Missing criterion: Some texts describe the development of revenue without ever mentioning revenue. Surrogate words like "market" may be used, but they are not discriminative enough.

Multiple criteria: Though we aimed at discarding sentences, in which revenue is mentioned without comprising a statement on it, in some cases our features did not work out, mainly due to intricate sentence structure.

Traps: Some sentences contain numeric values on revenue, but not the ones looked for, as in "10% of the revenue". We tackled these cases, but had still some false classifications left.

Hidden boundaries: Finally, we did not find all correct sentence boundaries, which can lead to both false positives and false negatives. The predominant problem was to separate headlines from paragraph beginnings and is partly caused by the missing access to markup tags.

5.4 Efficiency

We ran the identification algorithm on the whole corpus using a *2 GHz Intel Core 2 Duo MacBook* with *4 GB RAM*. The 1,128 corpus documents contain 33,370 sentences as counted by our algorithm itself. Tokenization, sentence splitting, time and money extraction took only 55.2 seconds, i.e., more than 20 documents or 600 sentences each second. Since our feature computation is not optimized yet, the complete identification process is a little less efficient with 7.35 documents or 218 sentences per second. However, it is fast enough to be used in online applications, which was our goal in the end.

Candidates	Data	Precision	Recall
Annotated	validation data	0.91	0.94
	test data	0.87	0.93
Extracted	validation data	0.90	0.89
	test data	0.86	0.87

Table 4: Precision and recall of the statement identification on manually annotated data and on automatically extracted data, respectively.

6 Conclusion

We presented a multi-stage approach for the automatic identification and aggregation of market statements and introduced a manually annotated German corpus for related tasks. The approach has been influenced by industry and is oriented towards practical applications, but is, in general, not specific to the German language. It relies on efficient retrieval, extraction and NLP techniques. By now, we can precisely identify most sentences that represent statements on revenue. This already allows for the support of strategists, e.g. by highlighting such sentences in web pages, which we currently implement as a Firefox extension. The overall problem is complex, though, and we are aware that human experts can do better at present. Nevertheless, time-consuming tasks can be automated and, in this respect, the results on our corpus are very promising.

Acknowledgement: This work was funded by the project "InfexBA" of the German Federal Ministry of Education and Research (BMBF) under contract number 01IS08007A.

References

Ahn, David, Sisay F. Adafre, and Maarten de Rijke. 2005. Extracting Temporal Information from Open Domain Text: A Comparative Exploration. *Journal of Digital Information Management*, 3(1): 14–20.

Berekoven, Ludwig, Werner Eckert, and Peter Ellenrieder. 2001. *Marktforschung: Methodische Grundlagen und praktische Anwendung*, 9th Edition, Gabler, Wiesbaden, Germany.

Carletta, Jean. 1996. Assessing Agreement on Classification Tasks: The Kappa Statistic. *Computational Linguistics*, 22: 249–254.

Cramer, Irene M., Stefan Schacht, and Andreas Merkel. 2007. Classifying Number Expressions in German Corpora. In *Proceedings of the 31st Annual Conference of the German Classification Society on Data Analysis, Machine Learning, and Applications*, pages 553–560.

Fan, Rong-En, Pai-Hsuen Chen, and Chih-Jen Lin. 2001. Working Set Selection Using Second Order Information for Training Support Vector Machines. *Journal of Machine Learning Research*, 6: 1889–1918.

Ferrucci, David and Adam Lally. 2004. UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment. *Natural Language Engineering*, 10(3–4): pages 327–348.

Glance, Natalie, Matthew Hurst, Kamal Nigam, Matthew Siegler, Robert Stockton, and Takashi Tomokiyo. 2005. Deriving Marketing Intelligence from Online Discussion. In *Proceedings of the Eleventh International Conference on Knowledge Discovery in Data Mining*, pages 419–428.

Gottron, Thomas. 2007. Evaluating Content Extraction on HTML Documents. In *Proceedings of the 2nd International Conference on Internet Technologies and Applications*, pages 123–132.

Hall, Mark, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11(1).

Hirao, Tsutomu, Hideki Isozaki, Eisaku Maeda and Yuji Matsumoto. 2002. Extracting Important Sentences with Support Vector Machines. In *Proceedings of the 19th International Conference on Computational linguistics*, pages 342–348.

Koppel, Moshe and Itai Shtrimberg. 2004. Good News or Bad News? Let the Market Decide. In *Proceedings of the AAAI Spring Symposium on Exploring Attitude and Affect in Text: Theories and Applications*, pages 86–88.

Lavrenko, Victor, Matt Schmill, Dawn Lawrie, Paul Ogilvie, David Jensen, and James Allan. 2000. Mining of Concurrent Text and Time Series. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining Workshop on Text Mining*, pages 37–44.

Lerman, Kevin, Ari Gilder, Mark Dredze, and Fernando Pereira. 2008. Reading the Markets: Forecasting Public Opinion of Political Candidates by News Analysis. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 473–480.

Liu, Yang, Xiangji Huang, Aijun An, and Xiaohui Yu. 2007. Arsa: A Sentiment-Aware Model for Predicting Sales Performance Using Blogs. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 607–614.

Mani, Inderjeet and George Wilson. 2000. Robust Temporal Processing of News. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 69–76.

Ratinov, Lev and Dan Roth. 2009. Design Challenges and Misconceptions in Named Entity Recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155.

Saquete, Estela, Rafael Muñoz, and Patricio Martínez-Barco. 2003. TERSEO: Temporal Expression Resolution System Applied to Event Ordering. *Text, Speech and Dialogue*, Springer, Berlin / Heidelberg, Germany, pages 220–228.

Stein, Benno, Sven Meyer zu Eissen, Gernot Gräfe, and Frank Wissbrock. 2005. Automating Market Forecast Summarization from Internet Data. *Fourth International Conference on WWW/Internet*, pages 395–402.

Steinwart, Ingo and Andreas Christmann. 2008. *Support Vector Machines*, Springer, New York, NY.

Xu, Jinxi and Bruce W. Croft 2000. Improving the effectiveness of information retrieval with local context analysis. *ACM Transactions on Information Systems*, 18(1): 79-112.

Towards a Unified Approach to Simultaneous Single-Document and Multi-Document Summarizations

Xiaojun Wan
Institute of Compute Science and Technology
The MOE Key Laboratory of Computational Linguistics
Peking University
wanxiaojun@icst.pku.edu.cn

Abstract

Single-document summarization and multi-document summarization are very closely related tasks and they have been widely investigated independently. This paper examines the mutual influences between the two tasks and proposes a novel unified approach to simultaneous single-document and multi-document summarizations. The mutual influences between the two tasks are incorporated into a graph model and the ranking scores of a sentence for the two tasks can be obtained in a unified ranking process. Experimental results on the benchmark DUC datasets demonstrate the effectiveness of the proposed approach for both single-document and multi-document summarizations.

1 Introduction

Single-document summarization aims to produce a concise and fluent summary for a single document, and multi-document summarization aims to produce a concise and fluent summary for a document set consisting of multiple related documents. The two tasks are very closely related in both task definition and solution method. Moreover, both of them are very important in many information systems and applications. For example, given a cluster of news articles, a multi-document summary can be used to help users to understand the whole cluster, and a single summary for each article can be used to help users to know the content of the specified article.

To date, single-document and multi-document summarizations have been investigated extensively and independently in the NLP and IR fields. A series of special conferences or workshops on automatic text summarization (e.g. SUMMAC, DUC, NTCIR and TAC) have advanced the technology and produced a couple of experimental online systems. However, the two summarization tasks have not yet been simultaneously investigated in a unified framework.

Inspired by the fact that the two tasks are very closely related and they can be used simultaneously in many applications, we believe that the two tasks may have mutual influences on each other. In this study, we propose a unified approach to simultaneous single-document and multi-document summarizations. The mutual influences between the two tasks are incorporated into a graph-based model. The ranking scores of sentences for single-document summarization and the ranking scores of sentences for multi-document summarization can boost each other, and they can be obtained simultaneously in a unified graph-based ranking process. To the best of our knowledge, this study is the first attempt for simultaneously addressing the two summarization tasks in a unified graph-based framework. Moreover, the proposed approach can be easily adapted for topic-focused summarizations.

Experiments have been performed on both the single-document and multi-document summarization tasks of DUC2001 and DUC2002. The results demonstrate that the proposed approach can outperform baseline independent methods for both the two summarization tasks. The two tasks are validated to have mutual influences on each other.

The rest of this paper is organized as follows: Section 2 introduces related work. The details of the proposed approach are described in Section 3. Section 4 presents and discusses the evaluation results. Lastly we conclude our paper in Section 5.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1137–1145,
Beijing, August 2010

2 Related Work

Document summarization methods can be either extraction-based or abstraction-based. In this section, we focus on extraction-based methods.

Extraction-based methods for single-document summarization usually assign a saliency score to each sentence in a document and then rank and select the sentences. The score is usually computed based on a combination of statistical and linguistic features, such as term frequency, sentence position, cue words and stigma words (Luhn, 1969; Edmundson, 1969; Hovy and Lin, 1997). Machine learning techniques have also been used for sentence extraction (Kupiec et al., 1995; Conroy and O'Leary, 2001; Shen et al., 2007; Li et al., 2009). The mutual reinforcement principle has been exploited to iteratively extract key phrases and sentences from a document (Zha, 2002; Wan et al, 2007a). Wan et al. (2007b) propose the CollabSum algorithm to use additional knowledge in a cluster of documents to improve single document summarization in the cluster.

In recent years, graph-based ranking methods have been investigated for document summarization, such as TextRank (Mihalcea and Tarau, 2004; Mihalcea and Tarau, 2005) and LexPageRank (ErKan and Radev, 2004). Similar to PageRank (Page et al., 1998), these methods first build a graph based on the similarity relationships between the sentences in a document and then the saliency of a sentence is determined by making use of the global information on the graph recursively. The basic idea underlying the graph-based ranking algorithm is that of "voting" or "recommendation" between sentences.

Similar methods have been used for generic multi-document summarization. A typical method is the centroid-based method (Radev et al., 2004). For each sentence, the method computes a score based on each single feature (e.g. cluster centroids, position and TFIDF) and then linearly combines all the scores into an overall sentence score. Topic signature is used as a novel feature for selecting important content in NeATS (Lin and Hovy, 2002). Various sentence features have been combined by using machine learning techniques (Wong et al., 2008). A popular way for removing redundancy between summary sentences is the MMR algorithm (Carbonell and Goldstein, 1998). Themes (or topics, clusters) in documents have been discovered and used for sentence selection (Harabagiu and Lacatusu, 2005). Hachey (2009) investigates the effect of various source document representations on the accuracy of the sentence extraction phase of a multi-document summarization task. Graph-based methods have also been used to rank sentences in a document set. The methods first construct a graph to reflect sentence relationships at different granularities, and then compute sentence scores based on graph-based learning algorithms. For example, Wan (2008) proposes to use only cross-document relationships for graph building and sentence ranking. Cluster-level information has been incorporated in the graph model to better evaluate sentences (Wan and Yang, 2008).

For topic-focused multi-document summarization, many methods are extensions of generic summarization methods by incorporating the information of the given topic or query into generic summarizers. In recent years, a few novel methods have been proposed for topic-focused summarization (Daumé and Marcu, 2006; Wan et al., 2007c; Nastase 2008; Li et al., 2008; Schilder and Kondadadi, 2008; Wei et al., 2008).

The above previous graph-based summarization methods aim to address either single-document summarization or multi-document summarization, and the two summarization tasks have not yet been addressed in a unified graph-based framework.

3 The Unified Summarization Approach

3.1 Overview

Given a document set, in which the whole document set and each single document in the set are required to be summarized, we use *local saliency* to indicate the importance of a sentence in a particular document, and use *global saliency* to indicate the importance of a sentence in the whole document set.

In previous work, the following two assumptions are widely made for graph-based summarization models:

Assumption 1: A sentence is locally important in a particular document if it is heavily linked with many locally important sentences in the same document.

Assumption 2: A sentence is globally important in the document set if it is heavily linked with many globally important sentences in the document set.

The above assumptions are the basis for PageRank-like algorithms for single document summarization and multi-document summarization, respectively. In addition to the above two assumptions, we make the following two assumptions to consider the mutual influences between the two summarization tasks:

Assumption 3: A sentence is locally important in a particular document, if it is heavily linked with many globally important sentences in the document set.

The above assumption is reasonable because the documents in the set are relevant and the globally important information in the document set will be expressed in many single documents. Therefore, if a sentence is salient in the whole document set, the sentence may be salient in a particular document in the set.

Assumption 4: A sentence is globally important in the document set, if it is heavily linked with many locally important sentences.

The above assumption is reasonable because the documents in the set are relevant and the globally important information in the whole set is the aggregation of the locally important information in each single document. Therefore, if a sentence is salient in a particular document, the sentence has the potential to be salient in the whole document set.

In brief, the local saliency and global saliency of a sentence can mutually influence and boost each other: high local saliency will lead to high global saliency, and high global saliency will lead to high local saliency.

Based on the above assumptions, our proposed approach first builds affinity graphs (each graph is represented by an affinity matrix) to reflect the different kinds of relationships between sentences, respectively, and then iteratively computes the local saliency scores and the global saliency scores of the sentences based on the graphs. Finally, the algorithm converges and the local saliency score and global saliency score of each sentence are obtained. The sentences with high local saliency scores in a particular document are chosen into the summary of the single document, and the sentences with high global saliency scores in the set are chosen into the summary of the document set.

Note that for both summarization tasks, after the saliency scores of sentences have been obtained, the greedy algorithm used in (Wan et al., 2007c) is applied to remove redundancy and finally choose both informative and novel sentences into the summary.

3.2 Algorithm Details

Formally, the given document set is denoted as $D=\{d_i|1\le i\le m\}$, and the whole sentence set is denoted as $S=\{s_i|1\le i\le n\}$. We let $Info_{single}(s_i)$ denote the local saliency score of sentence s_i in a particular document $d(s_i)\in D$, and it is used to select summary sentences for the single document $d(s_i)$. And we let $Info_{multi}(s_i)$ denote the global saliency score of sentence s_i in the whole document set D, and it is used to select summary sentences for the document set D.

The four assumptions in Section 3.1 can be rendered as follows:

$$Info_{\sin gle}(s_i) \propto \sum_j (W_A)_{ji} Info_{\sin gle}(s_j) \quad (1)$$

$$Info_{multi}(s_i) \propto \sum_j (W_B)_{ji} Info_{multi}(s_j) \quad (2)$$

$$Info_{\sin gle}(s_i) \propto \sum_j (W_C)_{ji} Info_{multi}(s_j) \quad (3)$$

$$Info_{multi}(s_i) \propto \sum_j (W_D)_{ji} Info_{\sin gle}(s_j) \quad (4)$$

where W_A, W_B, W_C, W_D are $n\times n$ affinity matrices reflecting the different kinds of relationships between sentences in the document set, where n is the number of all sentences in the document set. The detailed derivation of the matrices will be presented later.

After fusing the above equations, we can obtain the following unified forms:

$$\begin{aligned} Info_{\sin gle}(s_i) &= \mu \sum_j (W_A)_{ji} Info_{\sin gle}(s_j) \\ &\quad + (1-\mu)\sum_j (W_C)_{ji} Info_{multi}(s_j) \end{aligned} \quad (5)$$

$$\begin{aligned} Info_{multi}(s_i) &= \mu \sum_j (W_B)_{ji} Info_{multi}(s_j) \\ &\quad + (1-\mu)\sum_j (W_D)_{ji} Info_{\sin gle}(s_j) \end{aligned} \quad (6)$$

However, the above summarization method ignores the feature of sentence position, which has been validated to be very important for document summarizations. In order to incorporate this important feature, we add one prior score to each computation as follows:

$$\begin{aligned} Info_{\sin gle}(s_i) &= \alpha \sum_j (W_A)_{ji} Info_{\sin gle}(s_j) \\ &\quad + \beta \sum_j (W_C)_{ji} Info_{multi}(s_j) + \gamma \cdot prior_{\sin gle}(s_i) \end{aligned} \quad (7)$$

$$Info_{multi}(s_i) = \alpha \sum_j (W_B)_{ji} Info_{multi}(s_j) + \beta \sum_j (W_D)_{ji} Info_{\sin gle}(s_j) + \gamma \cdot prior_{multi}(s_i) \quad (8)$$

where α, β, $\gamma \in [0,1]$ specify the relative contributions to the final saliency scores from the different factors, and we have $\alpha+\beta+\gamma=1$. $prior_{single}(s_i)$ is the prior score for the local saliency of sentence s_i, and here $prior_{single}(s_i)$ is computed based on sentence position of s_i in the particular document $d(s_i)$. $prior_{multi}(s_i)$ is the prior score for the global saliency of sentence s_i, and we also compute $prior_{multi}(s_i)$ based on sentence position of s_i.

We use two column vectors $\vec{u} = [Info_{single}(s_i)]_{n\times 1}$ and $\vec{v} = [Info_{multi}(s_i)]_{n\times 1}$ to denote the local and global saliency scores of all the sentences in the set, respectively. And the matrix forms of the above equations are as follows:

$$\vec{u} = \alpha W_A^T \vec{u} + \beta W_C^T \vec{v} + \gamma \vec{p}_{\sin gle} \quad (9)$$

$$\vec{v} = \alpha W_B^T \vec{v} + \beta W_D^T \vec{u} + \gamma \vec{p}_{multi} \quad (10)$$

where $\vec{p}_{\sin gle} = [prior_{\sin gle}(s_i)]_{n\times 1}$ and $\vec{p}_{multi} = [prior_{multi}(s_i)]_{n\times 1}$ are the prior column vectors.

The above matrices and prior vectors are constructed as follows, respectively:

$\boldsymbol{W_A}$: This affinity matrix aims to reflect the local relationships between sentences in each single document, which is defined as follows:

$$(W_A)_{ij} = \begin{cases} sim_{\cos ine}(s_i, s_j), & \text{if } d(s_i) = d(s_j) \text{ and } i \neq j \\ 0, & \text{Otherwise} \end{cases} \quad (11)$$

where $d(s_i)$ refers to the document containing sentence s_i. $sim_{cosine}(s_i,s_j)$ is the cosine similarity between sentences s_i and s_j.

$$sim_{\cos ine}(s_i, s_j) = \frac{\vec{s}_i \cdot \vec{s}_j}{|\vec{s}_i| \times |\vec{s}_j|} \quad (12)$$

where $\vec{s}_i$ and $\vec{s}_j$ are the corresponding term vectors of s_i and s_j. Note that we have $(W_A)_{ij} = (W_A)_{ji}$, and we have $(W_A)_{ii} = 0$ to avoid self loops.

We can see that the matrix contains only the within-document relationships between sentences.

$\boldsymbol{W_B}$: This affinity matrix aims to reflect the global relationships between sentences in the document set, which is defined as follows:

$$(W_B)_{ij} = \begin{cases} sim_{\cos ine}(s_i, s_j), & \text{if } d(s_i) \neq d(s_j) \\ 0, & \text{Otherwise} \end{cases} \quad (13)$$

We can see that the matrix contains only the cross-document relationships between sentences. We do not include the within-document sentence relationships in the matrix because it has been shown that the cross-document relationships are more appropriate to reflect the global mutual influences between sentences than the within-document relationships in (Wan, 2008).

$\boldsymbol{W_C}$: This affinity matrix aims to reflect the cross-document relationships between sentences in the document set. However, the relationships in this matrix are used for carrying the influences of the sentences in other documents on the local saliency of the sentences in a particular document. If we directly use Equation (13) to compute the matrix, the mutual influences would be overly used. Because other documents might not be sampled from the same generative model as the specified document, we probably do not want to trust them so much as the specified document. Thus a confidence value is used to reflect out belief that the document is sampled from the same underlying model as the specified document. Heuristically, we use the cosine similarity between documents as the confidence value. And we use the confidence value as the decay factor in the matrix computation as follows:

$$(W_c)_{ij} = \begin{cases} sim_{\cos ine}(s_i, s_j) \times sim_{\cos ine}(d(s_i), d(s_j)), & \text{if } d(s_i) \neq d(s_j) \\ 0, & \text{Otherwise} \end{cases} \quad (14)$$

$\boldsymbol{W_D}$: This affinity matrix aims to reflect the within-document relationships between sentences. Thus we have $\boldsymbol{W_D}=\boldsymbol{W_A}$, which means that the global saliency score of a sentence is influenced only by the local saliency scores of the sentences in the same document, without considering the sentences in other documents.

Note that the above four matrices are symmetric and we can replace W_A^T, W_B^T, W_C^T and W_D^T by $\boldsymbol{W_A}$, $\boldsymbol{W_B}$, $\boldsymbol{W_C}$ and $\boldsymbol{W_D}$ in Equations (9) and (10), respectively.

$\boldsymbol{prior_{single}(s_i)}$: It is computed under the assumption that the first sentences in a document are usually more important than other sentences.

$$prior_{\sin gle}(s_i) = 0.5 + \frac{1}{position(s_i) + 1} \quad (15)$$

where $position(s_i)$ returns the position number of sentence s_i in its document $d(s_i)$. For example, if

s_i is the first sentence in its document, $position(s_i)$ is 1.

The prior weight is then normalized by:

$$prior_{single}(s_i) = \frac{prior_{single}(s_i)}{\sum_i prior_{single}(s_i)} \quad (16)$$

$prior_{multi}(s_i)$: We also let the prior weight reflect the influence of sentence position.

$$prior_{multi}(s_i) = prior_{single}(s_i) \quad (17)$$

And then the prior weight is normalized in the same way.

The above definitions are for generic document summarizations and the above algorithm can be easily adapted for topic-focused summarizations. Given a topic q, the only change for the above computation is $prior_{multi}(s_i)$. The topic relevance is incorporated into the prior weight as follows:

$$prior_{multi}(s_i) = sim_{cosine}(s_i, q) \quad (18)$$

$$prior_{multi}(s_i) = \frac{prior_{multi}(s_i)}{\sum_i prior_{multi}(s_i)} \quad (19)$$

In order to solve the iterative problem defined in Equations (9) and (10), we let $\vec{r} = [\vec{u}^T \ \vec{v}^T]^T$, $\vec{p} = [\gamma\vec{p}_{\text{single}}^T \ \gamma\vec{p}_{\text{multi}}^T]^T$, $\boldsymbol{W} = \begin{bmatrix} \alpha\boldsymbol{W}_A^T & \beta\boldsymbol{W}_C^T \\ \beta\boldsymbol{W}_D^T & \alpha\boldsymbol{W}_B^T \end{bmatrix}$, and then the iterative equations correspond to the following linear system:

$$\vec{r} = \boldsymbol{W}\vec{r} + \vec{p} \quad (20)$$

$$(\boldsymbol{I} - \boldsymbol{W})\vec{r} = \vec{p} \quad (21)$$

To guarantee the solution of the above linear system, $\boldsymbol{W}$ is normalized by columns. If all the elements of a column are zero, we replace the elements with $1/(2n)$, where $2n$ equals to the element number of the column. We then multiply $\boldsymbol{W}$ by a decay factor θ ($0<\theta<1$) to scale down each element in $\boldsymbol{W}$, but remain the meaning of $\boldsymbol{W}$. Here, θ is empirically set to 0.6[1]. Finally, Equation (21) is rewritten as follows:

$$(\boldsymbol{I} - \theta \cdot \boldsymbol{W})\vec{r} = \vec{p} \quad (22)$$

Thus, the matrix (I-$\theta\boldsymbol{W}$) is a strictly diagonally dominant matrix and the solution of the linear system exists and we can apply the Gauss-Seidel method used in (Li et al., 2008) to solve the linear system. The GS method is a well-know method for numeric computation in mathematics and the details of the method is omitted here.

[1] In our pilot study, we can observe good performance when θ is in a wide range of [0.4, 0.8].

4 Empirical Evaluation

4.1 Dataset and Evaluation Metric

Generic single-document and multi-document summarizations have been the fundamental tasks in DUC 2001 and DUC 2002 (i.e. tasks 1 and 2 in DUC 2001 and tasks 1 and 2 in DUC 2002), and we used the two datasets for evaluation. DUC2001 provided 309 articles, which were grouped into 30 document sets. Generic summary of each article was required to be created for task 1, and generic summary of each document set was required to be created for task 2. The summary length was 100 words or less. DUC 2002 provided 59 document sets consisting of 567 articles (D088 is excluded from the original 60 document sets by NIST) and generic summaries for each article and each document set with a length of approximately 100 words were required to be created. The sentences in each article have been separated and the sentence information has been stored into files. The summary of the two datasets are shown in Table 1.

	DUC 2001	**DUC 2002**
Task	Tasks 1, 2	Tasks 1, 2
Number of documents	309	567
Number of clusters	30	59
Data source	TREC-9	TREC-9
summary length	100 words	100 words

Table 1. Summary of datasets

We used the ROUGE toolkit[2] (Lin and Hovy, 2003) for evaluation, which has been widely adopted by DUC for automatic summarization evaluation. It measured summary quality by counting overlapping units such as the n-gram, word sequences and word pairs between the candidate summary and the reference summary.

The ROUGE toolkit reported separate recall-oriented scores for 1, 2, 3 and 4-gram, and also for longest common subsequence co-occurrences. We showed three of the ROUGE metrics in the experimental results: ROUGE-1 (unigram-based), ROUGE-2 (bigram-based), and ROUGE-W (based on weighted longest common subsequence, weight=1.2). In order to truncate summaries longer than the length limit,

[2] We used ROUGEeval-1.4.2 in this study.

we used the "-l 100" option in ROUGE toolkit. We also used the "-m" option for word stemming.

4.2 Evaluation Results

4.2.1 System Comparison

In the experiments, the combination weight γ for the prior score is fixed at 0.15, as in the PageRank algorithm. Therefore, we have $\alpha+\beta=0.85$. Here, we use $\alpha/(\alpha+\beta)$ to indicate the relative contributions of the first two parts in Equations (9) and (10). We empirically set $\alpha/(\alpha+\beta)=0.4$ in the experiments. The proposed unified approach (i.e. UnifiedRank) is compared with a few baseline approaches and the top three participating systems.

The graph-based baselines for single-document summarization are described as follows:

BasicRank: This baseline approach adopts the basic PageRank algorithm to rank sentences based on all sentence relationships in a single document, similar to previous work (Mihalcea and Tarau, 2004).

PositionRank: This baseline approach improves the basic PageRank algorithm by using the position weight of a sentence as the prior score for the sentence. The position weight of a sentence is computed by using Equation (15).

CollabRank1: This baseline approach is the "UniformLink(Gold)" approach proposed in (Wan et al. 2007b). It uses a cluster of multiple documents to improve single document summarization by constructing a global affinity graph.

CollabRank2: This baseline approach is the "UnionLink(Gold)" approach proposed in (Wan et al. 2007b).

The graph-based baselines for multi-document summarization are described as follows:

BasicRank: This baseline approach adopts the basic PageRank algorithm to rank sentences based on all sentence relationships in document set. Both within-document and cross-document sentence relationships are used for constructing the affinity graph.

PositionRank: Similarly, this baseline approach improves the basic PageRank algorithm by using the position weight of a sentence as the prior score for the sentence.

TwoStageRank: This baseline approach leverages the results of single document summarization for multi-document summarization. It first computes the score of each sentence within each single document by using the PositionRank method, and then computes the final score of each sentence within the document set by considering the document-level sentence score as the prior score in the improved PageRank algorithm.

The top three systems are the systems with highest ROUGE scores, chosen from the participating systems on each task, respectively. Tables 2 and 3 show the comparison results for single-document summarization on DUC2001 and DUC2002, respectively. Tables 4 and 5 show the comparison results for multi-document summarization on DUC2001 and DUC2002, respectively. In the tables, SystemX (e.g. System28, SystemN) represents one of the top performing systems. The systems are sorted by decreasing order of the ROUGE-1 scores.

For single-document summarization, the proposed UnifiedRank approach always outperforms the four graph-based baselines over all three metrics on both two datasets. The performance differences are all statistically significant by using t-test (p-value<0.05). The ROUGE-1 score of UnifiedRank is higher than that of the best participating systems and the ROUGE-2 and ROUGE-W scores of UnifiedRank are comparable to that of the best participating systems.

For multi-document summarization, the proposed UnifiedRank approach outperforms all the three graph-based baselines over all three metrics on the DUC2001 dataset, and it outperforms the three baselines over ROUGE-1 and ROUGE-W on the DUC2002 dataset. In particular, UnifiedRank can significantly outperform BasicRank and TwoStageRank over all three metrics on the DUC2001 dataset (t-test, p-value<0.05). Moreover, the ROUGE-1 and ROUGE-W scores of UnifiedRank are higher than that of the best participating systems and the ROUGE-2 score of UnifiedRank is comparable to that of the best participating systems.

The results demonstrate that the single-document and multi-document summarizations can benefit each other by making use of the mutual influences between the local saliency and

global saliency of the sentences. Overall, the proposed unified graph-based approach is effective for both single document summarization and multi-document summarization. However, the performance improvement for single-document summarization is more significant than that for multi-document summarization, which shows that the global information in a document set is very beneficial to summarization of each single document in the document set.

System	ROUGE-1	ROUGE-2	ROUGE-W
UnifiedRank	**0.45377**	**0.17649**	**0.14328**
CollabRank2	0.44038	0.16229	0.13678
CollabRank1	0.43890	0.16213	0.13676
PositionRank	0.43596	0.15936	0.13684
BasicRank	0.43407	0.15696	0.13629

Table 2. Comparison results for single-document summarization on DUC2001[3]

System	ROUGE-1	ROUGE-2	ROUGE-W
UnifiedRank	**0.48478**	0.21462	0.16877
System28	0.48049	**0.22832**	**0.17073**
System21	0.47754	0.22273	0.16814
CollabRank1	0.47187	0.20102	0.16318
CollabRank2	0.47028	0.20046	0.16260
PositionRank	0.46618	0.19853	0.16180
System31	0.46506	0.20392	0.16162
BasicRank	0.46261	0.19457	0.16018

Table 3. Comparison results for single-document summarization on DUC2002

System	ROUGE-1	ROUGE-2	ROUGE-W
UnifiedRank	**0.36360**	0.06496	**0.10950**
PositionRank	0.35733	0.06092	0.10798
BasicRank	0.35527	0.05608	0.10641
TwoStageRank	0.35221	0.05500	0.10515
SystemN	0.33910	0.06853	0.10240
SystemP	0.33332	0.06651	0.10068
SystemT	0.33029	**0.07862**	0.10215

Table 4. Comparison results for multi-document summarization on DUC2001

System	ROUGE-1	ROUGE-2	ROUGE-W
UnifiedRank	**0.38343**	0.07855	**0.12341**
PositionRank	0.38056	0.08238	0.12292
TwoStageRank	0.37972	0.08166	0.12261
BasicRank	0.37595	**0.08304**	0.12173
System26	0.35151	0.07642	0.11448
System19	0.34504	0.07936	0.11332
System28	0.34355	0.07521	0.10956

Table 5. Comparison results for multi-document summarization on DUC2002

[3] The summarization results for participating systems on DUC2001 are incomplete.

4.2.2 Influences of Combination Weight

In the above experiments, the relative contributions from the first two parts in Equations (9) and (10) are empirically set as $\alpha/(\alpha+\beta)=0.4$. In this section, we investigate how the relative contributions influence the summarization performance by varying $\alpha/(\alpha+\beta)$ from 0 to 1. A small value of $\alpha/(\alpha+\beta)$ indicates that the contribution from the same kind of saliency scores of the sentences is less important than the contribution from the different kind of saliency scores of the sentences, and vice versa. Figures 1-8 show the ROUGE-1 and ROUGE-W curves for single-document summarization and multi-document summarization on DUC2001 and DUC2002, respectively.

For single document summarization, very small value or very large value for $\alpha/(\alpha+\beta)$ will lower the summarization performance values on the two datasets. The results demonstrate that both the two kinds of contributions are important to the final performance of single document summarization.

For multi-document summarization, a relatively large value (≥ 0.4) for $\alpha/(\alpha+\beta)$ will lead to relatively high performance values on the DUC2001 dataset, but a very large value for $\alpha/(\alpha+\beta)$ will decrease the performance values. On the DUC2002 dataset, a relatively small value (≤ 0.4) will lead to relatively high performance values, but a very small value for $\alpha/(\alpha+\beta)$ will decrease the performance values. Though the trends of the curves on the DUC2001 and DUC2002 datasets are not very consistent with each other, the results show that both the two kinds of contributions are beneficial to the final performance of multi-document summarization.

5 Conclusion and Future Work

In this study, we propose a novel unified approach to simultaneous single-document and multi-document summarization by making using of the mutual influences between the two tasks. Experimental results on the benchmark DUC datasets show the effectiveness of the proposed approach.

In future work, we will perform comprehensive experiments for topic-focused document

summarizations to show the robustness of the proposed approach.

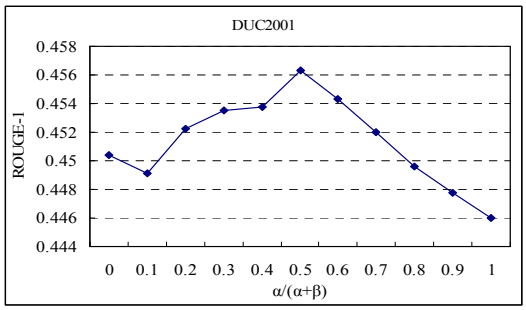


Figure 1. ROUGE-1 vs. combination weight for single-document summarization on DUC2001

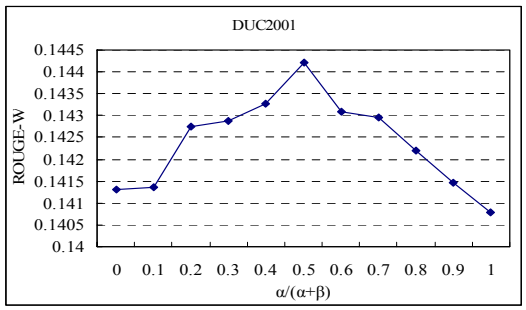


Figure 2. ROUGE-W vs. combination weight for single-document summarization on DUC2001

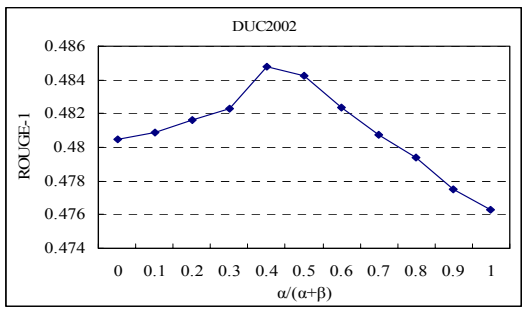


Figure 3. ROUGE-1 vs. combination weight for single-document summarization on DUC2002

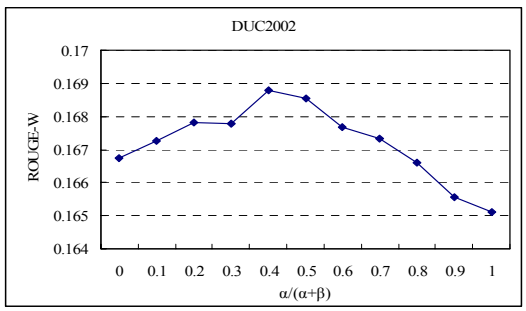


Figure 4. ROUGE-W vs. combination weight for single-document summarization on DUC2002

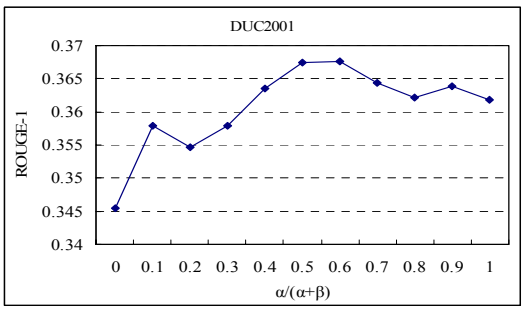


Figure 5. ROUGE-1 vs. combination weight for multi-document summarization on DUC2001

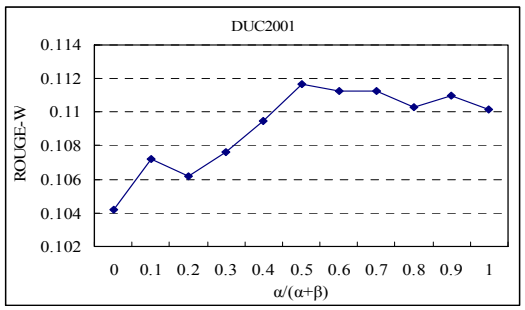


Figure 6. ROUGE-W vs. combination weight for multi-document summarization on DUC2001

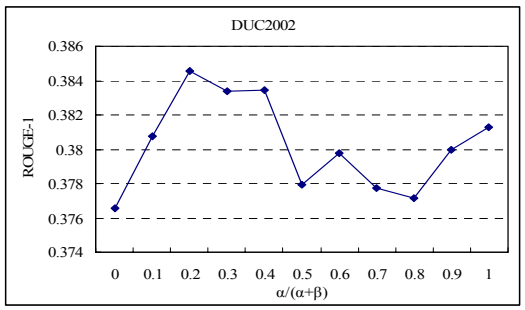


Figure 7. ROUGE-1 vs. combination weight for multi-document summarization on DUC2002

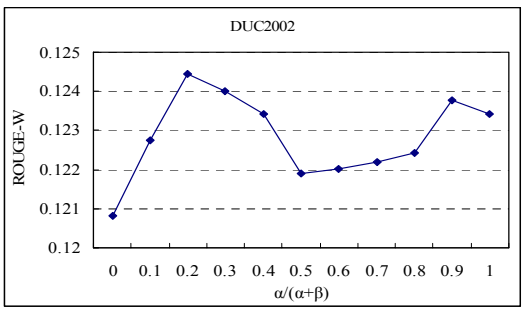


Figure 8. ROUGE-W vs. combination weight for multi-document summarization on DUC2002

Acknowledgments

This work was supported by NSFC (60873155), Beijing Nova Program (2008B03) and NCET (NCET-08-0006).

References

J. Carbonell, J. Goldstein. 1998. The Use of MMR, Diversity-based Reranking for Reordering Documents and Producing Summaries. In *Proceedings of SIGIR1998*, 335-336.

J. M. Conroy, D. P. O'Leary. 2001. Text Summarization via Hidden Markov Models. In *Proceedings of SIGIR2001*, 406-407.

H. Daumé and D. Marcu. 2006. Bayesian query-focused summarization. In *Proceedings of ACL-06*.

H. P. Edmundson. 1969. New Methods in Automatic Abstracting. *Journal of the Association for computing Machinery*, 16(2): 264-285.

G. ErKan, D. R. Radev. 2004. LexPageRank: Prestige in Multi-Document Text Summarization. In *Proceedings of EMNLP2004*.

B. Hachey. 2009. Multi-document summarisation using generic relation extraction. *In Proceedings of EMNLP2009*.

S. Harabagiu and F. Lacatusu. 2005. Topic themes for multi-document summarization. In *Proceedings of SIGIR-05*.

E. Hovy, C. Y. Lin. 1997. Automated Text Summarization in SUMMARIST. In *Proceeding of ACL'1997/EACL'1997 Worshop on Intelligent Scalable Text Summarization*.

J. Kupiec, J. Pedersen, F. Chen. 1995. A.Trainable Document Summarizer. In *Proceedings of SIGIR1995*, 68-73.

W. Li, F. Wei, Q. Lu and Y. He. 2008. PNR2: ranking sentences with positive and negative reinforcement for query-oriented update summarization. In *Proceedings of COLING-08*.

L. Li, K. Zhou, G.-R. Xue, H. Zha, Y. Yu. 2009. Enhancing diversity, coverage and balance for summarization through structure learning. In *Proceedings of WWW-09*.

C..-Y. Lin and E.. H. Hovy. 2002. From Single to Multi-document Summarization: A Prototype System and its Evaluation. In *Proceedings of ACL-02*.

C.-Y. Lin and E.H. Hovy. 2003. Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics. In *Proceedings of HLT-NAACL -03*.

H. P. Luhn. 1969. The Automatic Creation of literature Abstracts. *IBM Journal of Research and Development*, 2(2).

R. Mihalcea, P. Tarau. 2004. TextRank: Bringing Order into Texts. In *Proceedings of EMNLP2004*.

R. Mihalcea and P. Tarau. 2005. A language independent algorithm for single and multiple document summarization. In *Proceedings of IJCNLP-05*.

V. Nastase. 2008. Topic-driven multi-document summarization with encyclopedic knowledge and spreading activation. In *Proceedings of EMNLP-08*.

L. Page, S. Brin, R. Motwani, and T. Winograd. 1998. The pagerank citation ranking: Bringing order to the web. *Technical report*, Stanford Digital Libraries.

D. R. Radev, H. Y. Jing, M. Stys and D. Tam. 2004. Centroid-based summarization of multiple documents. *Information Processing and Management*, 40: 919-938.

F. Schilder and R. Kondadadi. 2008. FastSum: fast and accurate query-based multi-document summarization. In *Proceedings of ACL-08: HLT*.

D. Shen, J.-T. Sun, H. Li, Q. Yang, and Z. Chen. 2007. Document Summarization using Conditional Random Fields. In *Proceedings of IJCAI2007*.

X. Wan. 2008. Using Only Cross-Document Relationships for Both Generic and Topic-Focused Multi-Document Summarizations. *Information Retrieva*l, 11(1): 25-49.

X. Wan and J. Yang. 2008. Multi-document summarization using cluster-based link analysis. In *Proceedings of SIGIR-08*.

X. Wan, J. Yang and J. Xiao. 2007a. Towards an Iterative Reinforcement Approach for Simultaneous Document Summarization and Keyword Extraction. In *Proceedings of ACL2007*.

X. Wan, J. Yang and J. Xiao. 2007b. CollabSum: Exploiting Multiple Document Clustering for Collaborative Single Document Summarizations. In *Proceedings of SIGIR2007*.

X. Wan, J. Yang and J. Xiao. 2007c. Manifold-ranking based topic-focused multi-document summarization. In *Proceedings of IJCAI-07*.

F. Wei, W. Li, Q. Lu and Y. He. 2008. Query-sensitive mutual reinforcement chain and its application in query-oriented multi-document summarization. In *Proceedings of SIGIR-08*.

K.-F. Wong, M. Wu and W. Li. 2008. Extractive summarization using supervised and semi-supervised learning. In *Proceedings of COLING-08.*

H. Y. Zha. 2002. Generic Summarization and Keyphrase Extraction Using Mutual Reinforcement Principle and Sentence Clustering. In *Proceedings of SIGIR2002*, 113-120.

"Got You!": Automatic Vandalism Detection in Wikipedia with Web-based Shallow Syntactic-Semantic Modeling

William Yang Wang and Kathleen R. McKeown
Department of Computer Science
Columbia University
yw2347@columbia.edu kathy@cs.columbia.edu

Abstract

Discriminating vandalism edits from non-vandalism edits in Wikipedia is a challenging task, as ill-intentioned edits can include a variety of content and be expressed in many different forms and styles. Previous studies are limited to rule-based methods and learning based on lexical features, lacking in linguistic analysis. In this paper, we propose a novel Web-based shallow syntactic-semantic modeling method, which utilizes Web search results as resource and trains topic-specific n-tag and syntactic n-gram language models to detect vandalism. By combining basic task-specific and lexical features, we have achieved high F-measures using logistic boosting and logistic model trees classifiers, surpassing the results reported by major Wikipedia vandalism detection systems.

1 Introduction

Online open collaboration systems are becoming a major means of information sharing on the Web. With millions of articles from millions of resources edited by millions of people, Wikipedia is a pioneer in the fast growing, online knowledge collaboration era. Anyone who has Internet access can visit, edit and delete Wikipedia articles without authentication.

A primary threat to this convenience, however, is vandalism, which has become one of Wikipedia's biggest concerns (Geiger, 2010). To date, automatic countermeasures mainly involve rule-based approaches and these are not very effective. Therefore, Wikipedia volunteers have to spend a large amount of time identifying vandalized articles manually, rather than spending time contributing content to the articles. Hence, there is a need for more effective approaches to automatic vandalism detection.

In contrast to spam detection tasks, where a full spam message, which is typically 4K Bytes (Rigoutsos and Huynh, 2004), can be sampled and analyzed (Itakura and Clarke, 2009), Wikipedia vandals typically change only a small number of words or sentences in the targeted article. In our preliminary corpus (Potthast et al., 2007), we find the average size of 201 vandalized texts to be only 1K Byte. This leaves very few clues for vandalism modeling. The question we address in this paper is: given such limited information, how can we better understand and model Wikipedia vandalism?

Our proposed approach establishes a novel classification framework, aiming at capturing vandalism through an emphasis on shallow syntactic and semantic modeling. In contrast to previous work, we recognize the significance of natural language modeling techniques for Wikipedia vandalism detection and utilize Web search results to construct our shallow syntactic and semantic models. We first construct a baseline model that captures task-specific clues and lexical features that have been used in earlier work (Potthast et al., 2008; Smets et al., 2008) augmenting these with shallow syntactic and semantic features. Our main contributions are:

- Improvement over previous modeling methods with three novel lexical features
- Using Web search results as training data for syntactic and semantic modeling
- Building topic-specific n-tag syntax models and syntactic n-gram models for shallow syntactic and semantic modeling

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1146–1154,
Beijing, August 2010

2 Related Work

So far, the primary method for automatic vandalism detection in Wikipedia relies on rule-based bots. In recent years, however, with the rise of statistical machine learning, researchers have begun to treat Wikipedia vandalism detection task as a classification task. To the best of our knowledge, we are among the first to consider the shallow syntactic and semantic modeling using Natural Language Processing (NLP) techniques, utilizing the Web as corpus to detect vandalism.

ClueBot (Carter, 2007) is one of the most active bots fighting vandalism in Wikipedia. It keeps track of the IP of blocked users and uses simple regular expressions to keep Wikipedia vandalism free. A distinct advantage of rule-based bots is that they have very high precision. However they suffer from fixed-size knowledge bases and use only rigid rules. Therefore, their average recall is not very high and they can be easily fooled by unseen vandalism patterns. According to Smets et al., (2008) and Potthast et al., (2008), rule-based bots have a perfect precision of 1 and a recall of around 0.3.

The Wikipedia vandalism detection research community began to concentrate on the machine learning approaches in the past two years. Smets et al. (2008) wrapped all the content in *diff* text into a bag of words, disregarding grammar and word order. They used Naïve Bayes as the classification algorithm. Compared to rule-based methods, they show an average precision of 0.59 but are able to reach a recall of 0.37. Though they are among the first to try machine learning approaches, the features in their study are the most straightforward set of features. Clearly, there is still room for improvement.

More recently, Itakura and Clarke (2009) have proposed a novel method using Dynamic Markov Compression (DMC). They model their approach after the successful use of DMC in Web and Mail Spam detection (Bratko et al., 2006). The reported average precision is 0.75 and average recall is 0.73.

To the best of our knowledge, Potthast et al., (2008) report the best result so far for Wikipedia vandalism detection. They craft a feature set that consists of interesting task-specific features. For example, they monitor the number of previously submitted edits from the same author or IP, which is a good feature to model author contribution. Their other contributions are the use of a logistic regression classifier, as well as the use of lexical features. They successfully demonstrate the use of lexical features like vulgarism frequency. Using all features, they reach an average precision of 0.83 and recall of 0.77.

In addition to previous work on vandalism detection, there is also earlier work using the web for modeling. Biadsy et al. (2008) extract patterns in Wikipedia to generate biographies automatically. In their experiment, they show that when using Wikipedia as the only resource for extracting named entities and corresponding collocational patterns, although the precision is typically high, recall can be very low. For that reason, they choose to use Google to retrieve training data from the Web. In our approach, instead of using Wikipedia edits and historical revisions, we also select the Web as a resource to train our shallow syntactic and semantic models.

3 Analysis of Types of Vandalism

In order to better understand the characteristics of vandalism cases in Wikipedia, we manually analyzed 201 vandalism edits in the training set of our preliminary corpus. In order to concentrate on textual vandalism detection, we did not take into account the cases where vandals hack the image, audio or other multimedia resources contained in the Wikipedia edit.

We found three main types of vandalism, which are shown in Table 1 along with corresponding examples. These examples contain both the title of the edit and a snippet of the *diff*-ed content of vandalism, which is the textual difference between the old revision and the new revision, derived through the standard *diff* algorithm (Heckel, 1978).

- **Lexically ill-formed**
  This is the most common type of vandalism in Wikipedia. Like other online vandalism acts, many vandalism cases in Wikipedia involve ill-intentioned or ill-formed words such as vulgarisms, invalid letter sequences, punctuation misuse and Web slang. An interesting observation is that vandals almost never add emoticons in Wikipedia. For the first example in

Vandalism Types	Examples
Lexically ill-formed	**Edit Title**: *IPod* shit!!!!!!!!!!!!!!!!!!!!
Syntactically ill-formed	**Edit Title**: *Rock music* DOWN WITH SOCIETY MADDISON STREET RIOT FOREVER.
	Edit Title: *Vietnam War* Crabinarah sucks dont buy it
Lexically + syntactically well-formed, semantically ill-intentioned	**Edit Title**: *Global Warming* Another popular theory involving global warming is the concept that global warming is not caused by greenhouse gases. The theory is that Carlos Boozer is the one preventing the infrared heat from escaping the atmosphere. Therefore, the Golden State Warriors will win next season.
	Edit Title: *Harry Potter* Harry Potter is a teenage boy who likes to smoke crack with his buds. They also run an illegal smuggling business to their headmaster dumbledore. He is dumb!

Table 1: Vandalism Types and Examples

Table 1, vulgarism and punctuation misuse are observed.

- **Syntactically ill-formed**

  Most vandalism cases that are lexically ill-intentioned tend to be syntactically ill-formed as well. It is not easy to capture these cases by solely relying on lexical knowledge or rule-based dictionaries and it is also very expensive to update dictionaries and rules manually. Therefore, we think that is crucial to incorporate more syntactic cues in the feature set in order to improve performance. Moreover, there are also some cases where an edit could be lexically well-intentioned, yet syntactically ill-formed. The first example of syntactic ill-formed in Table 1 is of this kind.

Feature Sets	Features
Task-specific	Number of Revisions; Revisions Size Ratio;
Lexical	Vulgarism; Web Slang; Punctuation Misuse; Comment Cue Words;
Syntactic	Normalized Topic-specific N-tag Log Likelihood and Perplexity
Semantic	Normalized Topic-specific Syntactic N-gram Log Likelihood and Perplexity

Table 2: Feature Sets and Corresponding Features of Our Vandalism Detection System

- **Lexically and syntactically well formed, but semantically ill-intentioned**

  This is the trickiest type of vandalism to identify. Vandals of this kind might have good knowledge of the rule-based vandalism detecting bots. Usually, this type of vandalism involves off-topic comments, inserted biased opinions, unconfirmed information and lobbying using very subjective comments. However, a common characteristic of all vandalism in this category is that it is free of both lexical and syntactic errors. Consider the first example of semantic vandalism in Table 1 with edit title "Global Warming": while the first sentence for that edit seems to be fairly normal (the author tries to claim another explanation of the global warming effect), the second sentence makes a sudden transition from the previous topic to mention a basketball star and makes a ridiculous conclusion in the last sentence.

In this work, we realize the importance of incorporating NLP techniques to tackle all the above types of vandalism, and our focus is on the syntactically ill-formed and semantically ill-intentioned types that could not be detected by rule-based systems and straightforward lexical features.

4 Our System

We propose a shallow syntactic-semantic focused classification approach for vandalism detection (Table 2). In contrast to previous work, our approach concentrates on the aspect of using natural language techniques to model vandalism. Our shallow syntactic and semantic modeling approaches extend the traditional n-gram language modeling method with topic-specific n-tag (Collins et al., 2005) syntax models and topic-specific syntactic n-gram semantic models. Moreover, in the Wikipedia vandalism detection task, since we do not have a sufficient amount of training data to model the topic of each edit, we propose the idea of using the Web as corpus by retrieving search engine results to learn our topic-specific n-tag syntax and syntactic n-gram semantic models. The difference between our syntactic and semantic modeling is that n-tag syntax models only model the order of sentence constituents, disregarding the corresponding words. Conversely, for our syntactic n-gram models, we do keep track of words together with their POS tags and model both the word and syntactic compositions as a sequence. The detail of our shallow syntactic-semantic modeling method will be described in subsection 4.4.

We use our shallow syntactic-semantic model to augment our base model, which builds on early work. For example, when building one of our task-specific features, we extract the name of the author of this revision to query Wikipedia about the historical behavior of this author. This kind of task-specific global feature tends to be very informative and thus forms the basis of our system. For lexical level features, we count vulgarism frequencies and also introduce three new lexical features: Web slang, punctuation misuse and comment cue words, all of which will be described in detail in 4.2 and 4.3.

4.1 Problem Representation

The vandalism detection task can be formulated as the following problem. Let's assume we have a vandalism corpus *C*, which contains a set of Wikipedia edits *S*. A Wikipedia edit is denoted as $\boldsymbol{e}_i$. In our case, we have $S = \{\boldsymbol{e}_1, \boldsymbol{e}_2 \ldots, \boldsymbol{e}_n\}$. Each edit $\boldsymbol{e}$ has two consecutive revisions (an old revision R_{old} and a new revision R_{new}) that are unique in the entire data set. We write that $\boldsymbol{e} = \{R_{old}, R_{new}\}$. With the use of the standard *diff* algorithm, we can produce a text R_{diff}, showing the difference between these two revisions, so that $\boldsymbol{e} = \{R_{old}, R_{new}, R_{diff}\}$. Our task is: given S, to extract features from edit $\boldsymbol{e} \in S$ and train a logistic boosting classifier. On receiving an edit $\boldsymbol{e}$ from the test set, the classifier needs to decide whether this $\boldsymbol{e}$ is a vandalism edit or a non-vandalism edit. $\boldsymbol{e} \rightarrow \{1,0\}$.

4.2 Basic Task-specific and Lexical Features

Task-specific features are domain-dependent and are therefore unique in this Wikipedia vandalism detection task. In this work, we pick two task-specific features and one lexical feature that proved effective in previous studies.

- **Number of Revisions**
  This is a very simple but effective feature that is used by many studies (Wilkinson and Huberman, 2007; Adler et al., 2008; Stein and Hess, 2007). By extracting the author name for the new revision R_{new}, we can easily query Wikipedia and count how many revisions the author has modified in the history.

- **Revision Size Ratio**
  Revision size ratio measures the size of the new revision versus the size of the old revision in an edit. This measure is an indication of how much information is gained or lost in the new revision R_{new}, compared to the old revision R_{old}, and can be expressed as:

$$\text{RevRatio}(\boldsymbol{e}) = \frac{\sum_{w \in R\ new} \text{Count (w)}}{\sum_{w \in R\ old} \text{Count (w)}}$$

  where W represents any word token of a revision.

- **Vulgarism Frequency**
  Revision size ratio measures the size of the new revision versus the Vulgarism frequency was first introduced by Potthast et al. (2008). However, note that not all vulgarism words should be considered as vandalism and sometime even the Wikipedia edit's title and content themselves contain vulgarism words.

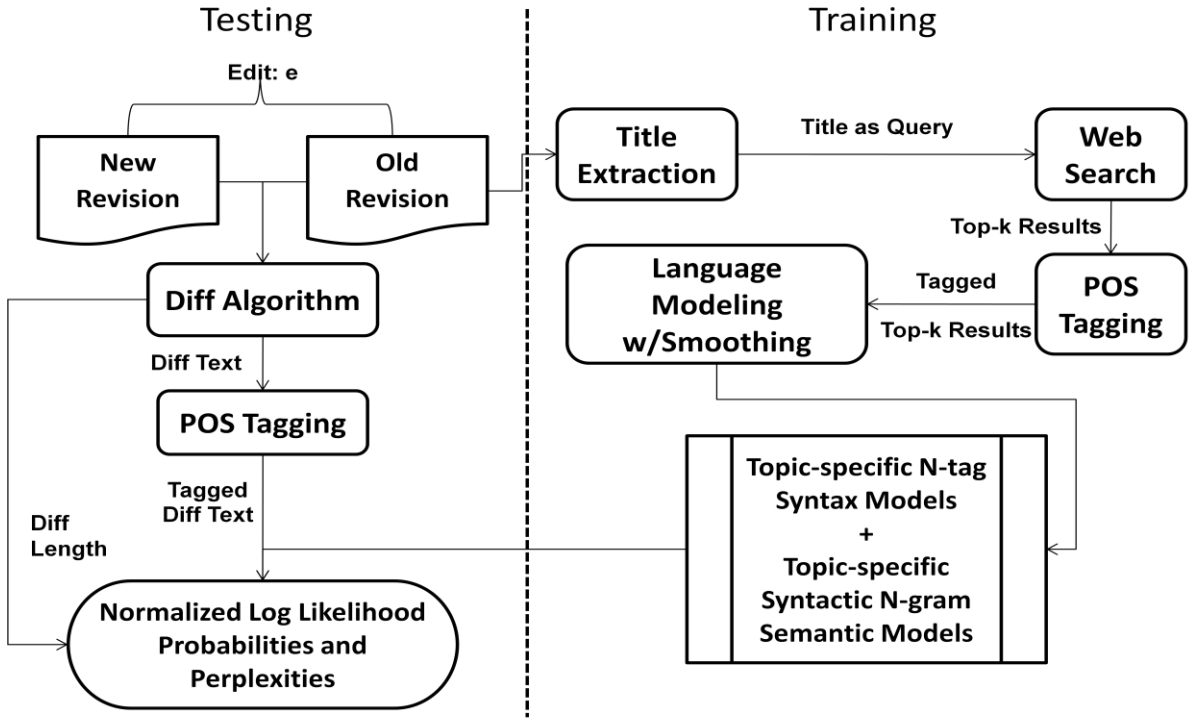


Figure 1. Topic-specific N-tag Syntax Models and Syntactical N-gram for Syntactical and Semantic Modeling

For each *diff* text in an edit ***e,*** we count the total number of appearances of vulgarism words *v* where *v* is in our vulgarism dictionary[1].

$$\text{VulFreq}(\boldsymbol{e}) = \sum_{v \in \text{Rdiff}} \text{Count}(v)$$

4.3 Novel Lexical Features

In addition to previous lexical features, we propose three novel lexical features in this paper: Web slang frequency, punctuation misuse, and comment cue words frequency.

- **Web Slang and Punctuation Misuse**

  Since Wikipedia is an open Web application, vandalism also contains a fair amount of Web slang, such as, "haha", "LOL" and "OMG". We use the same method as above to calculate Web slang frequency, using a Web slang dictionary[2]. In vandalism edits, many vandalism edits also contain punctuation misuse, for example, "!!!" and "???". However, we have not observed a significant amount of emoticons in the vandalism edits. Based on this, we only keep track of Web slang frequency and the occurrence of punctuation misuse.

- **Comment Cue Words**

  Upon committing each new revision in Wikipedia, the author is required to enter some comments describing the change. Well-intentioned Wikipedia contributors consistently use these comments to explain the motivation for their changes. For example, common non-vandalism edits may contain cue words and phrases like "edit revised, page changed, item cleaned up, link repaired or delinked". In contrast, vandals almost never take their time to add these kinds of comments. We can measure this phenomenon by counting the frequency of comment cue words.

[1] http://www.noswearing.com/dictionary
[2] http://www.noslang.com/dictionary/full

(a)
Rock/NNP and/CC roll/NN -LRB-/-LRB- also/RB spelled/VBD Rock/NNP 'n'/CC Roll/NNP
(b)
NNP CC NN -LRB- RB VBD NNP CC NNP
(c)
Rock/NNP !/. !/. !/. and/CC roll/VB you/PRP !/. !/. !/.
(d)
NNP . . . CC VB PRP . . .

Figure 2. Topic-specific N-tag and Syntactic N-gram modeling for the edit "Rock and Roll" in Wikipedia (a) The Web-derived POS tagged sequence (b) The Web-derived POS tag-only sequence (c) A POS tagged vandalism *diff* text R_{diff} (d) A POS tag-only vandalism R_{diff}

4.4 Topic-specific N-tag Syntax Models and Syntactic N-grams for Shallow Syntactic and Semantic Modeling

In Figure 1, we present the overview of our approach, which uses Web-trained topic-specific training for both: (1) n-tag syntax models for shallow syntactic modeling and (2) syntactic n-gram models for shallow semantic modeling.

For each Wikipedia edit, we consider its title as an approximate semantic representation, using it as a query to build topic-specific models. In addition, we also use the title information to model the syntax of this topic.

Given R_{diff}, we produce the syntactic version of the *diff*-ed text using a probabilistic POS tagger (Toutanova and Manning, 2000; Toutanova et al., 2003). The edit title is extracted from the corpus (either R_{new} or R_{old}) and is used to query multiple Web search engines in order to collect the n-tag and n-gram training data from the top-*k* results. Before we start training language models, we tag the top-*k* results using the POS tagger. Note that when modeling n-tag syntax models, it is necessary to remove all the words. With the POS-only sequences, we train topic-specific n-tag models to describe the syntax of normal text on the same topic associated with this edit. With the original tagged sequences, we train syntactic n-gram models to represent the semantics of the normal text of this edit.

After completing the training stage, we send the test segment (i.e. the *diff*-ed text sequence) to both the learned n-tag syntax models and the learned syntactic n-gram models. For the n-tag syntax model, we submit the POS tag-only version of the segment. For the syntactic n-gram model, we submit a version of the segment where each original word is associated with its POS-tag. In both cases we compute the log-likelihood and the perplexity of the segment.

Finally, we normalize the log likelihood and perplexity scores by dividing them by the length of R_{diff}, as this length varies substantially from one edit to another. [3] We expect an edit that has low log likelihood probability and perplexity to be vandalism, and it is very likely to be unrelated to the syntax and semantic of the normal text of this Wikipedia edit. In the end, the normalized log probability and perplexity scores will be incorporated into our back-end classifier with all task-specific and lexical features.

Web as Corpus: In this work, we leverage Web search results to train the syntax and semantic models. This is based on the assumption that the Web itself is a large corpus and Web search results can be a good training set to approximate the semantics and syntax of the query.

Topic-specific Modeling: We introduce a topic-specific modeling method that treats every edit in Wikipedia as a unique topic. We think that the title of each Wikipedia edit is an approximation of the topic of the edit, so we extract the title of each edit and use it as keywords to retrieve training data for our shallow syntactic and semantic modeling.

Topic-specific N-tag and Syntactic N-gram: In our novel approach, we tag all the top-*k* query results and *diff* text with a probabilistic POS tagger in both the training and test set of the vandalism corpus. Figure 2(a) is an example of a POS-tagged sequence in a top-*k* query result.

For shallow syntactic modeling, we use an n-tag modeling method (Collins et al., 2005). Given a tagged sequence, we remove all the words and only keep track of its POS tags: $tag_{i-2}\ tag_{i-1}$

[3] Although we have experimented with using the length of R_{diff} as a potential feature, it does not appear to be a good indicator of vandalism.

tag_i. This is similar to n-gram language modeling, but instead, we model the syntax using POS tags, rather than its words. In this example, we can use the system in Figure 2 (b) to train an n-tag syntactic model and use the one in Figure 2 (d) to test. As we see, for this test segment, it belongs to the vandalism class and has very different syntax from the n-tag model. Therefore, the normalized log likelihood outcome from the n-tag model is very low.

In order to model semantics, we use an improved version of the n-gram language modeling method. Instead of only counting $word_{i-2}$ $word_{i-1}$ $word_i$, we model composite tag/word feature, e.g. $tag_{i-2}word_{i-2}$ $tag_{i-1}word_{i-1}$ tag_iword_i. This syntactic n-gram modeling method has been successfully applied to the task of automatic speech recognition (Collins et al., 2005). In the example in Figure 2, the vandalism *diff* text will probably score low, because although it shares an overlap bigram "*and roll*" with the phrase "*rock and roll*" in training text, once we apply the shallow syntactic n-gram modeling method, the POS tag bigram "*and/CC roll/VB*" in *diff* text will be distinguished from the "*and/CC roll/NN*" or "*and/CC roll/NNP*" in the training data.

5 Experiments

To evaluate the effectiveness of our approach, we first run experiments on a preliminary corpus that is also used by previous studies and compare the results. Then, we conduct a second experiment on a larger corpus and analyze in detail the features of our system.

5.1 Experiment Setup

In our experiments, we use a Wikipedia vandalism detection corpus (Potthast et al., 2007) as a preliminary corpus. The preliminary corpus contains 940 human-assessed edits from which 301 edits are classified as vandalism. We split the corpus and keep a held-out 100 edits for each class in testing and use the rest for training. In the second experiment, we adopt a larger corpus (Potthast et al., 2010) that contains 15,000 edits with 944 marked as vandalism. The split is 300 edits for each class in held-out testing and the rest used for training. In the description of the second corpus, each edit has been reviewed by at least 3 and up to 15 annotators. If more than 2/3 of the annotators agree on a given edit, then the edit is tagged as one of our target classes. Only 11 cases are reported where annotators fail to form a majority inter-labeler agreement and in those cases, the class is decided by corpus authors arbitrarily.

In our implementation, the Yahoo![4] search engine and Bing[5] search engine are the source for collecting top-*k* results for topic-specific n-gram training data, because Google has a daily query limit. We retrieve top-100 results from Yahoo!, and combine them with the top-50 results from Bing.

For POS tagging, we use the Stanford POS Tagger (Toutanova and Manning, 2000; Toutanova et al., 2003) with its attached wsj3t0-18-bidirectional model trained from the Wall Street Journal corpus. For both shallow syntactic and semantic modeling, we train topic-specific trigram language models on each edit using the SRILM toolkit (Stolcke, 2002).

In this classification task, we used two logistic classification methods that haven't been used before in vandalism detection. Logistic model trees (Landwehr et al., 2005) combine tree induction with linear modeling. The idea is to use the logistic regression to select attributes and build logistic regression at the leaves by incrementally refining those constructed at higher levels in the tree. The second method we used, logistic boosting (Friedman et al., 2000), improves logistic regression with boosting. It works by applying the classification algorithm to reweighted versions of the data and then taking a weighted majority vote of the sequence of classifiers thus produced.

5.2 Preliminary Experiment

In the preliminary experiment, we tried logistic boosting classifiers and logistic model trees as classifiers with 10-fold cross validation. The rule-based method, ClueBot, is our baseline.

We also implemented another baseline system, using the bag of words (BoW) and Naive Bayes method (Smets et al., 2008) and the same toolkit (McCallum, 1996) that Smets et al. used. Then, we compare our result with Potthast et al. (2008), who used the same corpus as us.

[4] http://www.yahoo.com
[5] http://www.bing.com

Systems	Recall	Precision	F1
ClueBot	0.27	**1**	0.43
BoW + Naïve Bayes	0.75	0.74	0.75
Potthast et. al., 2008	0.77	0.83	0.80
Task-specific +Lexical (LMT)	0.87	0.87	0.87
Task-specific +Lexical (LB)	0.92	0.91	0.91
Our System (LMT)	0.89	0.89	0.89
Our System (LB)	**0.95**	0.95	**0.95**

Table 3: Preliminary Experiment Results; The acronyms: BoW: Bag of Words, LMT: Logistic Model Trees, LB: Logistic Boosting, Task-specific + Lexical: features in section 4.1 and 4.2

As we can see in Table 3, the ClueBot has a F-score (F1) of 0.43. The BoW + Naïve Bayes approach improved the result and reached an F1 of 0.75. Compared to these results, the system of Potthast et al. (2008) is still better and has a F1 of 0.80.

For the results of our system, LMT gives us a 0.89 F1 and LogitBoost (LB) gives a 0.95 F1. A significant F1 improvement of 15% was achieved in comparison to the previous study (Potthast et al., 2008). Another finding is that we find our shallow syntactic-semantic modeling method improves 2-4% over our task-specific and lexical features.

5.3 Results and Analysis

In the second experiment, a notable difference from the preliminary evaluation is that we have an unbalanced data problem. So, we use random down-sampling method to resample the majority class into balanced classes in the training stage. Then, we also use the two classifiers with 10-fold cross validation.

The F1 result reported by our BoW + Naïve Bayes baseline is 0.68. Next, we test our task-specific and lexical features that specified in section 4.1 and 4.2. The best result is a F1 of 0.82, using logistic boosting. Finally, with our topic-specific shallow syntactic and semantic modeling features, we have a precision of 0.86, a recall of 0.85 and F1 of 0.85.

Features	Recall	Precision	F1
BoW + Naïve Bayes	0.68	0.68	0.68
Task-specific (LMT)	0.81	0.80	0.80
Task-specific +Lexical(LMT)	0.81	0.81	0.81
Our System (LMT)	0.84	0.83	0.83
Task-specific (LB)	0.81	0.80	0.80
Task-specific + Lexical (LB)	0.83	0.82	0.82
Our System (LB)	**0.86**	**0.85**	**0.85**

Table 4: Second Experiment Results

Though we are surprised to see the overall F1 for the second experiment are not as high as the first one, we do see that the topic-specific shallow syntactic and semantic modeling methods play an important role in improving the result.

Looking back at the related work we mentioned in section 2, though we use newer data sets, our overall results still seem to surpass major vandalism detection systems.

6 Conclusion and Future Works

We have described a practical classification framework for detecting Wikipedia vandalism using NLP techniques and shown that it outperforms rule-based methods and other major machine learning approaches that are previously applied in the task.

In future work, we would like to investigate deeper syntactic and semantic cues to vandalism. We hope to improve our models using shallow parsing and full parse trees. We may also try lexical chaining to model the internal semantic links within each edit.

Acknowledgements

The authors are grateful to Julia Hirschberg, Yves Petinot, Fadi Biadsy, Mukund Jha, Weiyun Ma, and the anonymous reviewers for useful feedback. We thank Potthast et al. for the Wikipedia vandalism detection corpora.

References

Adler, B. Thomas, Luca de Alfaro, Ian Pye and Vishwanath Raman. 2008. Measuring Author Contributions to the Wikipedia. In *Proc. of the ACM 2008 International Symposium on Wikis*.

Biadsy, Fadi, Julia Hirschberg, and Elena Filatova. 2008. An Unsupervised Approach to Biography Production using Wikipedia. In *Proc. of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 807–815.

Bratko, Andrej, Gordon V. Cormack, Bogdan Filipic,Thomas R. Lynam and Blaz Zupan. 2006. Spam Filtering Using Statistical Data Compression Models. *Journal of Machine Learning Research*, pages 7:2673-2698.

Collins, Michael, Brian Roark and Murat Saraclar. 2005. Discriminative Syntactic Language Modeling for Speech Recognition. In *Proc. of the 43rd Annual Meeting of the Association for Computational Linguistics*. pages 507–514.

Friedman, Jerome, Trevor Hastie and Robert Tibshirani. 2000. Additive Logistic Regression: a Statistical View of Boosting. *Annals of Statistics* 28(2), pages 337-407.

Geiger, R. Stuart. 2010. The Work of Sustaining Order in Wikipedia: The Banning of a Vandal. In *Proc. of the 2010 ACM Conference on Computer Supported Cooperative Work*, pages 117-126.

Heckel, Paul. 1978. A Technique for Isolating Differences Between Files. *Communications of the ACM*, pages 264–268

Itakura, Kelly Y. and Charles L. A. Clarke. 2009. Using Dynamic Markov Compression to Detect Vandalism in the Wikipedia. In *Proc. of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 822-823.

Landwehr, Niels, Mark Hall and Eibe Frank. 2005. Logistic Model Trees. *Machine Learning*, 59(1-2), pages 161–205.

McCallum, Andrew. 1996. Bow: a Toolkit for Statistical Language Modeling, Text Retrieval, Classification and Clustering.

Potthast, Martin, Benno Stein, and Robert Gerling. 2008. Automatic Vandalism Detection in Wikipedia. In *Proc. of the 30th European Conference on Information Retrieval, Lecture Notes in Computer Science*, pages 663-668.

Potthast, Martin and Robert Gerling. 2007. Wikipedia Vandalism Corpus WEBIS-VC07-11. *Web Technology & Information Systems Group, Bauhaus University Weimar*.

Potthast, Martin, Benno Stein and Teresa Holfeld. 2010. PAN Wikipedia Vandalism Training Corpus PAN-WVC-10. *Web Technology & Information Systems Group, Bauhaus University Weimar*.

Rigoutsos, Isidore and Tien Huynh. 2004. Chung-Kwei: a pattern-discovery-based system for the automatic identification of unsolicited e-mail messages (SPAM). In *Proc. of the First Conference on E-mail and Anti-Spam.*

Smets, Koen, Bart Goethals and Brigitte Verdonk. 2008. Automatic Vandalism Detection in Wikipedia: Towards a Machine Learning Approach In *Proc. of AAAI '08, Workshop on Wikipedia and Artificial Intelligence*, pages 43-48.

Stein, Klaus and Claudia Hess. 2007. Does It Matter Who Contributes: a Study on Featured Articles in the German Wikipedia. In *Proc. of the ACM 18th Conference on Hypertext and Hypermedia*, pages 171–174.

Stolcke, Andreas. 2002. SRILM – an extensible language modeling toolkit. In *Proc. of the International Conference on Spoken Language Processing*, volume 2, pages 901–904.

Toutanova, Kristina and Christopher D. Manning. 2000. Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. In *Proc. of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 63-70.

Toutanova, Kristina, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *Proceedings of Human Language Technology Conference and the North American Chapter of the Association of Computational Linguistics Series*, pages 252-259.

Wilkinson,Dennis and Bernardo Huberman. 2007. Cooperation and Quality in Wikipedia. In *Proc. of the ACM 2007 International Symposium on Wikis*, pages 157–164.

Exploiting Salient Patterns for Question Detection and Question Retrieval in Community-based Question Answering

Kai Wang
Department of Computer Science
School of Computing
National University of Singapore
kwang@comp.nus.edu.sg

Tat-Seng Chua
Department of Computer Science
School of Computing
National University of Singapore
chuats@comp.nus.edu.sg

Abstract

Question detection serves great purposes in the cQA question retrieval task. While detecting questions in standard language data corpus is relatively easy, it becomes a great challenge for online content. Online questions are usually long and informal, and standard features such as question mark or 5W1H words are likely to be absent. In this paper, we explore question characteristics in cQA services, and propose an automated approach to detect question sentences based on lexical and syntactic features. Our model is capable of handling informal online languages. The empirical evaluation results further demonstrate that our model significantly outperforms traditional methods in detecting online question sentences, and it considerably boosts the question retrieval performance in cQA.

1 Introduction

Community-based Question Answering services (cQA) such as Yahoo! Answers have emerged as popular means of information exchange on the web. They not only connect a network of people to freely ask and answer questions, but also allow information seekers to search for relevant historical questions in the cQA archive (Agichtein et al., 2008; Xue et al., 2008; Wang et al., 2009).

Many research works have been proposed to find similar questions in cQA. The state-of-the-art retrieval models include the vector space model (Duan et al., 2008), language model (Duan et al., 2008; Jeon et al., 2005), Okapi model (Jeon et al., 2005), translation model (Jeon et al., 2005; Riezler et al., 2007; Xue et al., 2008), and syntactic tree matching model(Wang et al., 2009). Although experimental studies in these works show that the proposed models are capable of improving question retrieval, they did not give clear explanation on which portion of the question that the user query is actually matched against. A question thread from cQA usually comprises several sub-questions conveying different information needs, and it is highly desirable to identify individual sub-questions and match each of them to the user query. Getting sub-questions clearly identified not only helps the retrieval system to match user query to the most desirable content but also improves the retrieval efficiency.

However, the detection of sub-question is non-trivial. Question sentences in cQA are usually mixed with various description sentences, and they usually employ informal languages, where standard features such as question mark or utterance are likely to be absent. As such, simple heuristics using question mark or 5W1H words (*who, what, where, why, how*) may become inadequate. The demand of special techniques in detecting question sentences online arises due to three particular reasons. First, the question mark could be missing at the end of a question[1], or might be used in cases other than questions such as "*Really bad toothache?*". Second, some questions such as "*I'd like to know the expense of removing wisdom teeth*" are expressed in a declarative form, which neither contains 5W1H words nor is neccessarily ended with "?". Third, some question-like sentences do not carry any actual information need, such as "*Please help me?*". Figure 1 illustrates an example of a question thread

[1]It is reported (Cong et al., 2008) that 30% of online questions do not end with question marks.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1155–1163,
Beijing, August 2010

S1:	What do you guys do when you find that the 'plastic protection seal' is missing or disturbed.
S2:	Throw it out, buy a new one.. or just use it anyways?
S3:	Is it really possible or likely that the item you purchased was tampered with??
S4:	The box was in a plastic wrap but the item itself inside did not having the protection seal (box says it should) so I couldn't have inspected it before I bought it.
S5:	Please suggest?… thanks!

Figure 1: An example of a question thread extracted from Yahoo! Answers

from Yahoo! Answers, where sub-questions S1 and S2 are posted in non-standard forms, and S5 is merely a question-like simple sentence. To the best of our knowledge, none of the existing question retrieval systems are equipped with a comprehensive question detector module to handle various question forms online, and limited effort has been devoted to this direction.

In this paper, we extensively explore characteristics of questions in cQA, and propose a fully automated approach to detecting question sentences. In particular, we complement lexical patterns with syntactic patterns, and use them as features to train a classification model that is capable of handling informal online languages. To save human annotations, we further propose to employ one-class SVM algorithm for model learning, in which only positive examples are used as opposed to requiring both positive and negative examples.

The rest of the paper is organized as follows: Section 2 presents the lexical and syntactic patterns as used for question detection. Section 3 describes the learning algorithm for the classification model. Section 4 shows our experimental results. Section 5 reviews some related work and Section 6 concludes this paper.

2 Pattern Mining for Question Detection

As has been discussed, human generated content on the Web are usually not well formatted, and naive methods such as the use of question mark and 5W1H words are not adequate to correctly detect or capture all online questions. Methods based on hand-crafted rules also fail to cope with various question forms as randomly appeared on the Web. To overcome the shortcomings of these traditional methods, we propose to extract a set of salient patterns from online questions and use them as features to detect question sentences.

In this study, we mainly focus on two kinds of patterns – *sequential pattern* at the lexical level and *syntactic shallow pattern* at the syntactic level. Sequential patterns have been well discussed in many literature, including the identification of comparative sentences (Jindal and Liu, 2006), the detection of erroneous sentences (Sun et al., 2007) and question sentences (Cong et al., 2008) etc. However, works on syntactic patterns have only been partially explored (Zaki and Aggarwal, 2003; Sun et al., 2007; Wang et al., 2009). Grounded on these previous works, we next explain our mining approach of the sequential and syntactic shallow patterns.

2.1 Sequential Pattern Mining

Sequential Pattern is also referred to as *Labeled Sequential Pattern (LSP)* in the literature. It is in the form of $S \Rightarrow C$, where S is a sequence $\{t_1, \ldots, t_n\}$, and C is the class label that the sequence S is classified to. In the problem of question detection, a sequence is defined to be a series of tokens from questions, and the class labels are $\{Q, NQ\}$, which stand for question and non-question respectively.

The purpose of sequential pattern mining is to extract a set of frequent subsequence of words that are indicative of questions. For example, the word subsequence "*anyone know what ... to*" could be a good indication to characterize the question sentence "*anyone know what I can do to make me less tired.*". Note that the mined sequential tokens need not to be contiguous as appeared in the original text.

There is a handful of algorithms available for frequent subsequence extraction. Pei et al. (2001) observed that all occurrences of a frequent pattern can be classified into groups (approximated pattern) and proposed a Prefixspan algorithm. The Prefixspan algorithm quickly finds out all relative frequent subsequences by a pattern growth method, and determines the approximated patterns from those subsequences. We adopt this algorithm in our work due to its high reported efficiency. We impose the following additional constraints for better control over the significance of the mined patterns:

1. Maximum Pattern Length: It limits the maximum number of tokens in a mined sequence.
2. Maximum Token Distance: The two adjacent tokens t_n and t_{n+1} in the pattern need to be within a threshold window in the original text.
3. Minimum Support: The minimum percentage of sentences in Q containing the pattern p.
4. Minimum Confidence: The probability of a pattern $p \Rightarrow Q$ being true in the whole database.

To overcome the word sparseness problem, we generalize each sentence by applying the Part-of-Speech (POS) tags to all tokens except some indicative keywords such as 5W1H words, modal words, stopwords etc. For instance, the question sentence *"How can I quickly tell if my wisdom teeth are coming"* is converted to *"How can I RB VBP if my NN NNS VBP VBG"*, on top of which the pattern mining is conducted. To further capture online language patterns, we mine a set of frequent tokens that are unique to cQA such as *"any1"*, *"im"* and *"whats"*, and keep them from being generalized. The reason to hold back this set of tokens is twofold. First, conventional POS taggers are trained from standard English corpus, and they could mis-tag these non-standard words. Second, the special online tokens are analogue to standard stopwords, and having them properly excluded could help reflect the online users' textual questioning patterns.

It is expected that the converted patterns preserve the most representative features of online questions. Each discovered pattern makes up a binary feature for the classification model that we will introduce in Section 3.

2.2 Syntactic Shallow Pattern Mining

The sequential patterns represent features at the lexical level, but we found that lexical patterns might not always be adequate to categorize questions. For example, the pattern $\{$*when, do*$\}$ could presume the non-question "*Levator scapulae is used when you do the traps workout*" to be a question, whereas the question "*know someone with an eating disorder?*" could be overlooked due to the lack of indicative lexical patterns.

These limitations, however, could be alleviated by syntactic features. The syntactic pattern *(SBAR(WHADVP(WRB))(S(NP)(VP)))* extracted

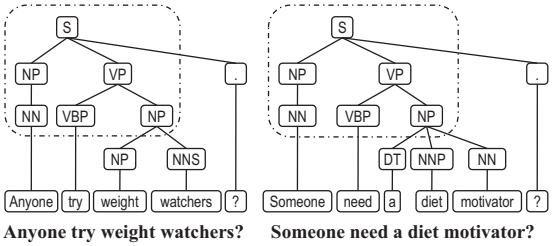


Figure 2: An example of common syntactic patterns observed in two different question sentences

from the former example has the order of *NP* and *VP* being switched, which could indicate the sentence to be a non-question, whereas the pattern *(VP(VB)(NP(NP)(PP)))* may be evidence that the latter example is indeed a question, because this pattern is commonly witnessed in the archived questions. Figure 2 shows an example that two questions bear very different wordings but share the same questioning pattern *(S(NP(NN))(VP(VPB)(NP)))* at the syntactic level. In view of the above, we argue that patterns at the syntactic level could complement lexical patterns in identifying question sentences.

To our knowledge, the mining of salient patterns at the syntactic level was limited to a few tasks. Zaki and Aggarwal (2003) employed tree patterns to classify XML data, Sun et al. (2007) extracted all frequent sub-tree structures for erroneous sentences detection, and Wang et al. (2009) decomposed the parsing tree into fragments and used them to match similar questions. Our work differs from these previous works in that: (1) we also utilize syntactic patterns for the question detection; and (2) we do not blindly extract all possible sub-tree structures, but focus only on certain portions of the parsing tree for better pattern representation and extraction efficiency.

Given a syntactic tree T, we define *syntactic pattern* as a part of sub-structures of T such that the production rule for each non-leaf node in the patterns is intact. For example, the pattern *(S(NP(NN))(VP(VPB)(NP)))* in Figure 2 is considered to be a valid syntactic pattern, whereas *(S(NP(NN))(VP(VPB)))* is not, since the production rule *VP→VPB·NP* is not strictly complied.

We take the following measures to mine salient syntactic patterns: First, we limit the depth of each syntactic pattern to be within a certain range.

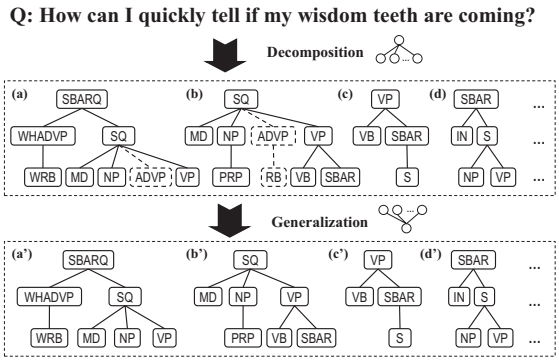


Figure 3: Illustration of syntactic pattern extraction and generalization process

It is believed that the syntax structure will become too specific if it is extended to a deeper level or too general if the depth is too shallow, neither of which produces good representative patterns. We therefore set the depth D of each syntactic pattern to be within a reasonable range ($2 \leq D \leq 4$). Second, we prune away all leaf nodes as well as the production rules at the POS tag level. We believe that nodes at the bottom levels do not carry much useful structural information favored by question detector. For example, the simple grammar rule *NP→DT·NN* does not give any insight to useful question structures. Third, we relax the definition of syntactic pattern by allowing the removal of some nodes denoting modifiers, preposition phrases, conjunctions etc. The reason is that these nodes are not essential in representing the syntactic patterns and are better excluded for generalization purpose. Figure 3 gives an illustration of the process for pattern extraction and generalization. In this example, several syntactic patterns are generated from the question sentence "*How can I quickly tell if my wisdom teeth are coming?*", and the tree patterns (a) and (b) are generalized into (a') and (b'), in which the redundant branch *(ADVP(RB))* that represents the adverb "*quickly*" is detached.

Contents on the Web are prone to noise, and most off-the-shelf parsers are not well-trained to parse online questions. For example, the parsing tree of the question "*whats the matter with it?*" will be very different from that of the question "*what is the matter with it?*". It would certainly be nice to know that "*whats*" is a widely used short form of the phrase "*what is*" on the Web, but we are lack of this kind of thesaurus. Nevertheless, we argue that the parsing errors would not hurt the question detector performance much as long as the mining database is large enough. The reason is that if certain irregular forms frequently occur on the Web, there will be statistical evidences that the syntactic patterns derived from it, though not desired, will commonly occur as well. In other words, we take the wrong patterns and utilize them to detect questions in the irregular forms. Our approach differs from other systems in that we do not intentionally try to rectify the grammatical errors, but leave the errors as they are and use the statistical based approach to capture those informal patterns.

The pattern extraction process is outlined in Algorithm 1. The overall mining strategy is analogous to the mining of sequential patterns, where *support* and *confidence* measures are taken into account to control the significance of the mined patterns. All mined syntactic patterns together with the lexical patterns will be used as features for learning the classification model.

Algorithm 1 *ExtractPattern(S, D)*

Input: A set of syntactic trees for sentences (S); the depth range (D)
Output: A set of sub-tree patterns extracted from S

```
Patterns = {}
for all Syntactic tree T ∈ S do
    Nodes ← Top-down level order traversal of T
    for all node n ∈ Nodes do
        Extract subtree p rooted under node n, with depth
        within the range D
        p ← generalize(p)
        Patterns.add(p)
    end for
end for
return Patterns
```

3 Learning the Classification Model

Although Conditional Random Fields (CRF) is good sequential learning algorithm and has been used in other related work (Cong et al., 2008), here we select Support Vector Machines (SVM) as an alternative learner. The reason is that our task not only deals with sequential patterns but also involves syntactic patterns that possess no sequential criteria. Additionally, SVM has been widely shown to provide superior results compared to other classifiers.

The input to a SVM binary classifier normally consists of both positive and negative examples. While it is easy to discover certain patterns from questions, it is unnatural to identify characteristics for non-questions, as they usually do not share such common lexical and syntactic patterns. The lack of good negative examples leads traditional SVM to perform poorly. To adapt the imbalanced input data, we proposed to employ a one-class SVM method (Manevitz and Yousef, 2002) for learning. The basic idea of one-class SVM is to transform features from only positive examples via a kernel to a hyper-plane and treats the origin as the only member of the second class. It uses relaxation parameters to separate the positive examples from the origin, and finally applies the standard two-class SVM techniques to learn a decision boundary. As a result, anything outside the boundary are considered to be outliers (*i.e.* non-questions in this problem).

More formally, given n training samples $x_1, \ldots, x_n$ of one class, the hyperplane separating them from the origin is constructed by solving

$$\min \frac{1}{2}\|w\|^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i - \rho \qquad (1)$$

subject to: $w \cdot \Phi(x_i) \geq \rho - \xi_i$, where Φ is a kernel function, ξ_i is the slack variable, and ν is the parameter controlling the upper bound percentage of outliers. If w and ρ solve this problem, the decision function $f(x) = sign(w \cdot \Phi(x) - \rho)$ will be positive for most examples x_i in the training set.

Supervised learning methods usually require training data to be manually annotated. To save labeling efforts, we take a shortcut by treating all sentences ending with question marks as an initial positive examples. This assumption is acceptable, as Cong et al. (2008) reported that the rule-based method using only question mark achieves a very high precision of over 97% in detecting questions. It in turn indicates that questions ending with "?" are highly reliable to be real questions.

However, the initial training data still contain many sentences ending with "?" but are not true questions. These possible outliers will shift the decision boundary away from the optimal one, and we need to remove them from the training dataset for better classification. Many preprocessing strategies are available for training data refinement, including bootstrapping, condensing, and editing etc. In this work, we employ a SVM-based data editing and classification method proposed by Song et al. (2008), which iteratively sets a small value to the parameter ν of the one-class SVM so as to continuously refine the decision boundary. The algorithm could be better visualized with Figure 4. In each iteration, a new decision boundary will be determined based on the existing set of data points, and a portion of possible outliers will be removed from the training set. It is expected that the learned hyperplane will eventually be very close to the optimal one.

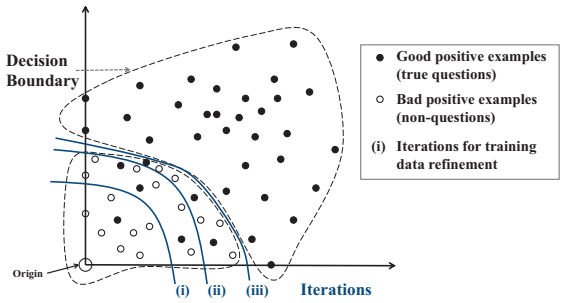


Figure 4: Illustration of one-class SVM classification with training data refinement (conceptual only). Three iterations (i) (ii) (iii) are presented.

We use the freely available software LIBSVM[2] to conduct the one-class SVM training and testing. A linear kernel is used, as it is shown to be superior in our experiments. In each refinement iteration, the parameter ν is conservatively set to 0.02. The number of iteration is dynamically determined according to the algorithm depicted in (Song et al., 2008). Other parameters are all set to default. The refined decision boundary from the training dataset will be applied to classify questions from non-questions. The question detector model learned will serve as a component for the cQA question retrieval system in our experiments.

4 Experiments

In this section, we present empirical evaluation results to assess the effectiveness of our question detection model. In particular, we first examine the effects of the number of patterns on question detection performance. We further conduct experiments to show that our question de-

[2] Available at: http://www.csie.ntu.edu.tw/~cjlin/libsvm

# of Lexical Patterns		Confidence					# of Syntactic Patterns		Confidence				
		60%	65%	70%	75%	80%			60%	65%	70%	75%	80%
Support	0.40%	1685	1639	1609	1585	1545	Support	0.03%	916	758	638	530	453
	0.45%	1375	1338	1314	1294	1277		0.04%	707	580	488	402	341
	0.50%	1184	1151	1130	1113	1110		0.05%	546	450	375	308	261
	0.55%	1037	1007	989	975	964		0.06%	468	379	314	260	218

Table 1: Number of lexical and syntactic patterns mined over different *support* and *confidence* values

Lexical Patterns		Confidence									Syntactic Patterns		Confidence								
		65%			70%			75%					60%			65%			70%		
		P	R	F_1	P	R	F_1	P	R	F_1			P	R	F_1	P	R	F_1	P	R	F_1
Support	0.40%	85.7	90.7	88.1	86.9	88.6	87.7	87.8	86.6	87.2	Support	0.03%	80.4	83.3	81.9	85.1	77.5	81.1	90.7	70.2	79.1
	0.45%	86.6	90.2	88.4	88.9	88.5	88.7	89.6	86.7	88.2		0.04%	79.0	86.1	82.4	90.1	78.2	83.7	90.8	70.8	79.6
	0.50%	88.5	91.6	88.4	86.4	89.0	87.7	86.2	87.9	87.0		0.05%	80.3	82.5	81.4	88.8	78.4	83.3	89.9	69.0	78.1
	0.55%	86.5	89.9	88.1	88.1	87.5	87.8	88.0	89.2	88.6		0.06%	83.0	83.2	83.1	88.5	77.2	82.4	86.7	75.8	80.9

Table 2: Question detection performance over different sets of lexical patterns and syntactic patterns

tection model combining both lexical and syntactic features outperforms traditional rule-based or lexical-based methods. We finally demonstrate that our question detection model gives additional performance boosting to question matching.

4.1 Performance Variation over Different Pattern Sets

The performance of the question detection model can be sensitive to the number of features used for learning. To find the optimal number of features used for model training, we examine the performance variation over different amount of lexical and syntactic patterns undertaken for training.

Dataset: We collected a total of around 800k question threads from Yahoo! Answers Healthcare domain. From the collected data, we generated the following three datasets:

- Pattern Mining Set: Comprising around 350k sentences from 60k question threads, where those ending with "?" are treated as questions and others as non-questions.
- Training Set: Positive examples comprising around 130k sentences ending with "?" from another 60k question threads for the one-class SVM learning algorithm.
- Testing Set: Two annotators are asked to tag randomly picked sentences from the remaining set. A total of 2,004 question sentences and 2,039 non-question sentences are annotated.

Methods & Results: We use different combinations of *support* and *confidence* values to generate different set of patterns. The *support* value ranges from 0.40% to 0.55% for lexical patterns with a step size of 0.05%, and ranges from 0.03% to 0.06% for syntactic patterns with a step size of 0.01%. The *confidence* value for both patterns ranges from 60% to 80% with a step size of 5%. These value ranges are empirically determined. Table 1 presents the number of lexical and syntactic patterns mined against different *support* and *confidence* value combinations.

For each set of lexical or syntactic patterns mined, we use them as features for model training. We convert the training sentences into a set of feature vectors and employ the one-class SVM algorithm to train a classifier. The classifier will then be applied to predict the question sentences in the testing set. To evaluate each question detection model, we employ Precision (P), Recall (R), and F_1 as performance metrics, and Table 2 presents the results[3].

We observe from Table 2 that given a fixed support level, the precision generally increases with the confidence level for both lexical and syntactic patterns, but the recall drops. The lexical feature set comprising 1,314 sequential patterns as generated with {*sup=0.45%*, *conf=70%*} gives the best F_1 score of 88.7%, and the syntactic feature set comprising 580 syntactic patterns generated from {*sup=0.04%*, *conf=65%*} gives the best F_1 score of 83.7%. It is noted that the sequential patterns give relatively high recall while the syntactic patterns give relatively high precision. Our reading is that the sequential patterns are capable of capturing most questions, but it may also give wrong predictions to non-questions such as "*Lev-*

[3]The results for certain *confidence* levels are not very promising and are not shown in the table due to lack of space.

ator scapulae is used when you do the traps workout" that bears the sequential pattern $\{$*when, do*$\}$. On the other hand, the syntactic patterns could give reliable predictions, but its coverage could suffer due to the limited number of syntactic patterns. We conjecture that a combination of both features could further improve the performance.

4.2 Performance Comparison with Traditional Question Detection Methods

We next conduct experiments to compare the performance of our question detection model to traditional rule-based or lexical-based methods.

Methods & Results: We set up five different systems for meaningful comparisons:

1. 5W1H (baseline1): a rule-based method using 5W1H to determine a question sentence.
2. Question Mark (baseline2): a method using the question mark "?" to judge a question.
3. SeqPattern: Using only the set of 1,314 sequential patterns as features.
4. SynPattern: Using only the set of 580 syntactic patterns as features.
5. SeqPattern+SynPattern: Merging both lexical and syntactic patterns and use them as a set of features for question detection.

We again employ Precision (P), Recall (R), and F_1 as performance metrics to evaluate each question detection system, and tabulate the comparison results in Table 3. From the Table, we observe that 5W1H performs poorly in both precision and recall, and question mark based method gives relatively low recall although the precision is the highest amongst all the methods evaluated. This is in line with the results as observed in (Cong et al., 2008). SeqPattern outperforms the two baseline systems in both R and F_1 scores, and its combination with SynPattern augments the performance in both precision and recall by a lot. It also achieves statistically significant improved results (t-test, p-value$<$0.05) as compared to other four systems. These results are consistent with our intuition that syntactic patterns can leverage sequential patterns in improving the question detection performance.

It is noted that SeqPattern+SynPattern exhibits the highest recall (R) amongst all the systems. The significance test further suggests that many question sentences miss-detected by 5W1H or Question Mark method could be properly captured by our model. This improvement is meaningful, as the question coverage is also an important factor in the cQA question retrieval task, where high recall implies that more similar questions could be matched and returned, hence improving the question retrieval performance.

System Combination	$P(\%)$	$R(\%)$	$F_1(\%)$
(1) 5W1H	75.37	49.50	59.76
(2) Question Mark	**94.12**	77.50	85.00
(3) SeqPattern	88.92	88.47	88.69
(4) SynPattern	90.06	78.19	83.71
(5) SeqPattern+SynPattern	92.11	**89.67**	**90.87**

Table 3: Performance comparisons for question detection on different system combinations

4.3 Performance Evaluation on Question Retrieval with Question Detection Model

To further demonstrate that our question detection model can improve question retrieval, we incorporate it into different question retrieval systems.

Methods: We select a simple bag-of-word (BoW) system retrieving questions at the lexical level, and a syntactic tree matching (STM) model matching questions at the syntactic level (Wang et al., 2009) as two baselines. For each baseline, we further set up two different combinations:

- Baseline+QM: Using question mark to detect question sentences, and perform question retrieval on top of the detected questions.
- Baseline+QD: Using our proposed model to detect question sentences, and perform question retrieval on top of the detected questions.

This gives rise to additional 4 different system combinations for comparison.

Dataset: We divide the dataset from Yahoo! Answers into a question repository set (750k) and a test set (50k). For the baseline systems, all the repository sentences containing both questions and non-questions are indexed, whereas for systems equipped with QM or QD, only the detected question sentences are indexed for retrieval. We randomly select 250 single-sentence questions from the test set as queries, and for each query, the retrieval system will return a list of top 10 question matches. We combine the retrieved results from different systems and ask two annotators to label each result to be either "relevant" or "irrel-

System Combination	BoW	BoW +QM	BoW +QD	STM	STM +QM	STM +QD
MAP (%)	58.07	59.89	60.68	66.53	68.41	69.85
% improvement of MAP over:						
Baseline	N.A.	+3.13	+4.49	N.A.	+2.83	+4.99
Baseline+QM	N.A.	N.A.	+1.32	N.A.	N.A.	+2.10
P@1 (%)	59.81	61.21	63.55	63.08	64.02	65.42

Table 4: Question retrieval performance on different system combinations measured by MAP and P@1 (Baseline is either BoW or STM)

evant" without telling them which system the result is generated from. By eliminating some query questions that have no relevant matches, the final testing set contains 214 query questions.

Metrics & Results: We evaluate the question retrieval performance using two metrics: Mean Average Precision (MAP) and Top One Precision (P@1). The results are presented in Table 4.

We can see from Table 4 that STM outperforms BoW. Applying QM or QD over BoW and STM boosts the system performance in terms of both MAP and P@1. They also achieve statistical significance as judged by paired t-test (p-value$<$0.05). More specifically, the MAP on QM coupled systems improves by 3.13% and 2.83% respectively over BoW and STM. This is evidence that having question sentences clearly identified could help to retrieve relevant questions more precisely, as without question detection, the user query is likely to be matched to irrelevant description sentences. Our question detection model (QD) further improves the MAP by 1.32% and 2.1% respectively over BoW+QM and STM+QM, and it also yields better top one precision by correctly retrieving questions at the first position on 136 and 140 questions respectively, out of a total of 214 questions. These improvements are in line with our expectation that our model incorporating salient features at both the lexical and syntactic levels is comprehensive enough to capture various forms of questions online, and hence improve the performance of question matching.

5 Related Work

Research on detecting question sentences can generally be classified into two categories. The first category simply employs rule-based methods such as question mark, 5W1H words, or handcrafted regular expressions to detect questions. As discussed, these conventional methods are not adequate to cope with online questions.

The second category uses machine learning approaches to detect question sentences. Shrestha and McKeown (2004) proposed a supervised rule induction method to detect interrogative questions in email conversations based on part-of-speech features. Yeh and Yuan (2003) used a statistical approach to extract a set of question-related words and derived some syntax and semantic rules to detect mandarin question sentences. Cong et al. (2008) extracted labeled sequential patterns and used them as features to learn a classifier for question detection in online forums.

Question pattern mining is also closely related to the learning of answer patterns. Work on answer patterns includes the web based pattern mining (Zhang and Lee, 2002; Du et al., 2005) and a combination of syntactic and semantic elements (Soubbotin and Soubbotin, 2002) etc.

In contrast to previous work, we do not only focus on standard language corpus, but extensively explore characteristics of online questions. Our approach exploits salient question patterns at both the lexical and syntactic levels for question detection. In particular, we employ the one-class SVM algorithm such that the learning process is weakly supervised and no human annotation is involved.

6 Conclusion

This paper proposed a new approach to detecting question sentences in cQA. We mined both lexical and syntactic question patterns, and used them as features to build classification models. The mining and leaning process is fully automated and requires no human intervention. Empirical evaluation on the cQA archive demonstrated the effectiveness of our model as well as its usefulness in improving question retrieval performance.

We are still investigating other features that are helpful to detect questions. One promising direction for future work is to also employ lexical and syntactic patterns to other related areas such as question type classification etc. It is also interesting to employ a hybrid of CRF and SVM learning methods to boost the accuracy and scalability of the classifier.

References

Agichtein, Eugene, Carlos Castillo, Debora Donato, Aristides Gionis, and Gilad Mishne. 2008. Finding high-quality content in social media. In *WSDM*, pages 183–194.

Cong, Gao, Long Wang, Chin-Yew Lin, Young-In Song, and Yueheng Sun. 2008. Finding question-answer pairs from online forums. In *SIGIR*, pages 467–474.

Du, Yongping, Helen Meng, Xuanjing Huang, and Lide Wu. 2005. The use of metadata, web-derived answer patterns and passage context to improve reading comprehension performance. In *HLT*, pages 604–611.

Duan, Huizhong, Yunbo Cao, Chin-Yew Lin, and Yong Yu. 2008. Searching questions by identifying question topic and question focus. In *HLT-ACL*, pages 156–164.

Jeon, Jiwoon, W. Bruce Croft, and Joon Ho Lee. 2005. Finding similar questions in large question and answer archives. In *CIKM*, pages 84–90.

Jindal, Nitin and Bing Liu. 2006. Identifying comparative sentences in text documents. In *SIGIR*, pages 244–251.

Manevitz, Larry M. and Malik Yousef. 2002. One-class svms for document classification. *J. Mach. Learn. Res.*, 2:139–154.

Pei, Jian, Jiawei Han, Behzad Mortazavi-asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei chun Hsu. 2001. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *ICDE*, pages 215–224.

Riezler, Stefan, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu Mittal, and Yi Liu. 2007. Statistical machine translation for query expansion in answer retrieval. In *ACL*, pages 464–471.

Shrestha, Lokesh and Kathleen McKeown. 2004. Detection of question-answer pairs in email conversations. In *COLING*, page 889.

Song, Xiaomu, Guoliang Fan, and M. Rao. 2008. Svm-based data editing for enhanced one-class classification of remotely sensed imagery. *Geoscience and Remote Sensing Letters, IEEE*, 5(2):189–193.

Soubbotin, Martin M. and Sergei M. Soubbotin. 2002. Use of patterns for detection of likely answer strings: A systematic approach. In *TREC*.

Sun, Guihua, Gao Cong, Xiaohua Liu, Chin-Yew Lin, and Ming Zhou. 2007. Mining sequential patterns and tree patterns to detect erroneous sentences. In *AAAI*, pages 925–930.

Wang, Kai, Zhaoyan Ming, and Tat-Seng Chua. 2009. A syntactic tree matching approach to finding similar questions in community-based qa services. In *SIGIR*, pages 187–194.

Xue, Xiaobing, Jiwoon Jeon, and W. Bruce Croft. 2008. Retrieval models for question and answer archives. In *SIGIR*, pages 475–482.

Yeh, Ping-Jer and Shyan-Ming Yuan. 2003. Mandarin question sentence detection: A preliminary study. In *EPIA*, pages 466–478.

Zaki, Mohammed J. and Charu C. Aggarwal. 2003. Xrules: an effective structural classifier for xml data. In *KDD*, pages 316–325.

Zhang, Dell and Wee Sun Lee. 2002. Web based pattern mining and matching approach to question answering. In *TREC*.

Probabilistic Tree-Edit Models with Structured Latent Variables for Textual Entailment and Question Answering

Mengqiu Wang
Computer Science Department
Stanford University
mengqiu@cs.stanford.edu

Christopher D. Manning
Computer Science Department
Stanford University
manning@cs.stanford.edu

Abstract

A range of Natural Language Processing tasks involve making judgments about the semantic relatedness of a pair of sentences, such as Recognizing Textual Entailment (RTE) and answer selection for Question Answering (QA). A key challenge that these tasks face in common is the lack of explicit alignment annotation between a sentence pair. We capture the alignment by using a novel probabilistic model that models tree-edit operations on dependency parse trees. Unlike previous tree-edit models which require a separate alignment-finding phase and resort to ad-hoc distance metrics, our method treats alignments as structured latent variables, and offers a principled framework for incorporating complex linguistic features. We demonstrate the robustness of our model by conducting experiments for RTE and QA, and show that our model performs competitively on both tasks with the same set of general features.

1 Introduction

Many complex Natural Language Processing (NLP) applications can be broken down to a subtask of evaluating the semantic relationship of pairs of sentences (e.g., in Question Answering, answer selection involve comparing each answer candidate against the question). This means that research aiming at analyzing pairs of semantically related natural language sentences is promising because of its reusability: it is not tied to a particular internal representation of meanings, but it nevertheless serves as a first step towards full meaning understanding, which is applicable to a number of applications. At the same time, this paradigm clearly defines the input and output space, facilitating system comparison and standard evaluation. Tasks of this paradigm have drawn much of the focus in recent NLP research, including Recognizing Textual Entailment (RTE), answer selection for Question Answering (QA), Paraphrase Identification (PI), Machine Translation Evaluation (MTE), and many more.

In each of these tasks, inputs to the systems are pairs of sentences that may or may not convey the desired semantic property (e.g., in RTE, whether the hypothesis sentence can be entailed from the premise sentence; in QA, whether the answer candidate sentence correctly answers the question), and the output of the system is a binary classification decision (or a regression score,as in MTE).

Earlier studies in these domains have concluded that simple word overlap measures (e.g., bag of words, n-grams) have a surprising degree of utility (Papineni et al., 2002; Jijkoun and de Rijke, 2005b), but are nevertheless not sufficient for these tasks (Jijkoun and de Rijke, 2005a). A common problem identified in these earlier systems is the lack of understanding of the semantic relation between words and phrases. Later systems that include more linguistic features extracted from resources such as WordNet have enjoyed more success (MacCartney et al., 2006). Studies have also shown that certain prominent syntactic features are often found beneficial (Snow et al., 2006). More recent studies gained further leverage from systematic exploration of the syntactic feature space through analysis of parse trees (Wang et al.,

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1164–1172,
Beijing, August 2010

2007; Das and Smith, 2009).

There are two key challenges imposed by these tasks. The first challenge has to do with the hidden alignment structures embedded in the sentence pairs. It is straightforward to see that in order to extract word-matching and/or syntax-matching features, inevitably one has to consider the alignment between words and/or syntactic parts. These alignments are not given as inputs, and it is a non-trivial task to decide what the *correct* alignment is. Alignment-based approach have been proven effective by many RTE, QA and MTE systems (Haghighi et al., 2005; Wang et al., 2007; MacCartney et al., 2008; Das and Smith, 2009, *inter alia*). Although alignment is a commonly used approach, it is not the only one. Other studies have successfully applied theorem proving and logical induction techniques, translating both sentences to knowledge representations and then doing inference on these representations (Moldovan et al., 2003; Raina et al., 2005; de Salvo Braz et al., 2005; MacCartney and Manning, 2007, *inter alia*).

A second challenge arises when a system needs to combine various sources of evidence (i.e., surface text features, semantic features, and syntactic features) to make a global classification decision. Quite often these features are heavily overlapping and sometimes contradicting, and thus a robust learning scheme that knows when to activate what feature is desired. Traditional approaches employ a two-stage or multi-stage model where tasks are broken down into alignment finding, feature extraction, and feature learning subtasks (Haghighi et al., 2005; MacCartney et al., 2008). The alignment finding task is typically done by committing to a *one best* alignment, and subsequent features are extracted only according to this alignment. A large body of literature in joint learning has demonstrated that such an approach can suffer from cascaded errors at testing, and does not benefit from the potential for joint learning (Finkel et al., 2006).

In this paper, we present a novel undirected graphical model to address these challenges. A promising approach to these challenges is modeling the alignment as an edit operation sequence over parse tree representation, an approach pioneered by (Punyakanok et al., 2004; Kouylekov and Magnini, 2006; Harmeling, 2007; Mehdad, 2009). We improve upon this earlier work by showing how alignment structures can be inherently learned as structured latent variables in our model. Tree edits are represented internally as state transitions in a Finite-State Machine (FSM), and our model is parameterized as a Conditional Random Field (CRF) (Lafferty et al., 2001), which allows us to incorporate a diverse set of arbitrarily overlapping features.

In comparison to previous work that exploits various ad-hoc or heuristic ways of incorporating tree-edit operations, our model provides an elegant and much more principled way of describing tree-edit operations in a probabilistic setting.

2 Tree-edit CRF for Classification

A training instance consists of a pair of sentences and an associated binary judgment. In RTE, for example, the input sentence pairs is made up of a text sentence (e.g., *Gabriel Garcia Marquez is a novelist and winner of the Nobel prize for literature.*) and a hypothesis sentence (e.g., *Gabriel Garcia Marquez won the Nobel for Literature.*). The pair is judged to be *true* if the hypothesis can be entailed from the text (e.g., the answer is *true* for the example sentence pair).

Formally, we denote the text sentence as *txt* and the hypothesis sentence as *hyp*, and denote their labeled dependency parse trees as $\tau_{\mathbf{t}}$ and $\tau_{\mathbf{h}}$, respectively. We use the binary variable $z \in \{0,1\}$ to denote the judgment.

The generative story behind our model is a parse tree transformation process. $\tau_{\mathbf{t}}$ is transformed into $\tau_{\mathbf{h}}$ through a sequence of *tree edits*. Examples of tree edits are *delete child*, *insert parent*, and *substitute current*. An edit sequence $\mathbf{e} = e_1 \ldots e_m$ is *valid* if $\tau_{\mathbf{t}}$ can be successfully turned into $\tau_{\mathbf{h}}$ according to $\mathbf{e}$. An example of a trivial valid edit sequence is one that first *deletes* all nodes in $\tau_{\mathbf{t}}$ then *inserts* all nodes in $\tau_{\mathbf{h}}$.

Delete, *insert* and *substitute* form the three basic edit operations. Each step in an edit sequence is also linked with current edit positions in both trees, denoted as $\mathbf{e}.\mathbf{p} = e_1.p \ldots e_m.p$. We index the tree nodes using a level-order tree traversal scheme (i.e., root is visited first and assigned in-

dex 0, then each one of the first level children of the root is visited in turn, and assigned an index number incremented by 1). It is worth noting that every valid edit sequence has a corresponding alignment mapping. Nodes that are inserted or deleted are aligned to *null*, and nodes that are substituted are aligned. One can find many edit sequence for the same alignment, by altering the order of edit operations.

We extend these basic edit operations into more elaborate edit operations based on the linguistic and syntactic properties of the current tree nodes that they fire on. For example, the following are all possible edit operations: *delete* a noun that is SUB of the root, *delete* a named-entity of type PERSON, *substitute* roots of the tree. In our experiments, we designed a set of 45 edit operations (12 *delete*, 12 *insert* and 21 *substitute*). More details of the edit operations are described in §4. Depending on the specific application domain, more sophisticated and verbose tree edit operations can be designed and easily incorporated into our model. In particular, tree edit operations involving deleting, inserting or substituting entire treelets seem interesting and promising, requiring merely a simple extension to the forward-backward dynamic programming.

Next, we design a Finite-State Machine (FSM) in which each edit operation is mapped to a unique state, and an edit sequence is mapped into a transition sequence among states (denoted as $\mathbf{e}.\mathbf{a} = e_1.a \ldots e_m.a$). In brief, an edit sequence is associated with a sequence of edit positions in the trees ($\mathbf{e}.\mathbf{p} = e_1.p \ldots e_m.p$), as well as a transition sequence among states ($\mathbf{e}.\mathbf{a} = e_1.a \ldots e_m.a$).

The probability of an edit sequence $\mathbf{e}$ given the parse trees is defined as:

$$P(\mathbf{e} \mid \tau_\mathbf{t}, \tau_\mathbf{h}) = \frac{1}{Z} \prod_{i=1}^{|\mathbf{e}|} \exp \theta \cdot \mathbf{f}(e_{i-1}, e_i, \tau_\mathbf{t}, \tau_\mathbf{h}) \quad (1)$$

where $\mathbf{f}$ are feature functions, θ are associated feature weights, and Z is the partition function to be defined next.

Recall that our training data is composed of not only positive examples but also negative examples. In order to take advantage of this label information, we adopt an interesting discriminative learning framework first introduced by McCallum et al. (2005). We call the FSM state set described above the *positive* state set ($\mathcal{S}_1$), and duplicate the exact same set of states, and call the new set *negative* state set ($\mathcal{S}_0$). We then add a starting state(S_s), and add non-deterministic transitions from S_s to every state in $\mathcal{S}_1$. We then add the same transitions for $\mathcal{S}_0$. We now arrive at a new FSM structure where upon arriving at the starting state, one makes a non-deterministic decision to enter either the positive set or the negative set and stay in that set until reaching the end of the edit sequence, since no transitions are allowed across the positive and negative set. Each edit operation sequence can now be associated with a sequence of positive states as well as a sequence of negative states. The intuitive idea is that during training, we want to maximize the weights of the positive examples in the positive state set and minimize their weights in the negative state set, and vice versa. In other words, we want the positive state set to attract positive examples but push away negative examples. Figure 1 illustrates two example valid edit sequences in the FSM, one in the positive state set and one in the negative state set.

Formally, the partition function Z in (1) is defined as the sum of weights of all valid edit sequences in both the positive set and negative set. Features extracted from positive states are disjoint from features extracted from negative states.

$$Z = \sum_{\mathbf{e}: \mathbf{e}.\mathbf{a} \subseteq S_s + \{\mathcal{S}_0 \bigcup \mathcal{S}_1\}^*} \prod_{i=1}^{|\mathbf{e}|} \exp \theta \cdot \mathbf{f}(e_{i-1}, e_i, \tau_\mathbf{t}, \tau_\mathbf{h})$$

Recall $z \in \{0, 1\}$ is the binary judgment indicator variable. The conditional probability of z is obtained by marginalizing over all edit sequences that have state transitions in the state set corresponding to z:

$$P(z \mid \tau_\mathbf{t}, \tau_\mathbf{h}) = \sum_{\mathbf{e}: \mathbf{e}.\mathbf{a} \subseteq S_s + \mathcal{S}_z^*} P(\mathbf{e} \mid \tau_\mathbf{t}, \tau_\mathbf{h}) \quad (2)$$

The L_2-norm penalized log-likelihood over n training examples ($\mathcal{L}$) is our training objective function:

$$\mathcal{L} = \sum_{j=1}^{n} \log(P(z^{(j)} \mid \tau_\mathbf{t}^{(j)}, \tau_\mathbf{h}^{(j)})) - \frac{\|\theta\|^2}{2\sigma^2} \quad (3)$$

At test time, the z with higher probability is taken as our prediction outcome.

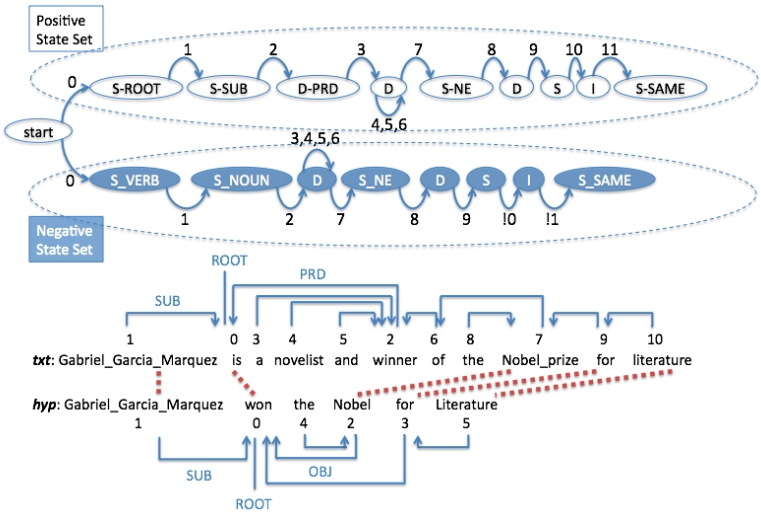


Figure 1: This diagram illustrates the FSM architecture. There is a single start state, and we can transit into either the positive state set (nodes that are not shaded), or the negative state set (shaded nodes). Here we show two examples of valid edit sequences. They result in the same alignment structure as show in the bottom half of the diagram (dotted lines across the two sentences are alignment links). Numbers over the arcs in the state diagram denote the edit sequence index, and numbers under each word in the parse tree diagram denote each node's level-order index number.

3 Parameter Estimation

We used Expectation Maximization method since the objective function given in (3) is non-convex. In the M-step, finding the optimal parameters under the current model expectation involves computing forward-backward style dynamic programming (DP) in a three-dimensional table (two for inputs and one for states) and optimization using L-BFGS method. In practice the resulting DP table can be quite large (for a sentence pair of length 100, and 2 sets of 45 states, we obtain 900,000 entries). We improved efficiency by pruning out partial sequences that do not lead to a complete valid sequence and pre-compute the state-transition table and features.

4 Edit Operations

Table 1 lists the groups of edit operations we designed and their descriptions. Not shown in the table are three default edits (*insert*, *delete* and *substitute*), which fire when none of the more specific edit operations match. Edit operations listed in the the top-left section capture basic matching, deletion and insertion of surface text, part-of-speech tags and named-entity tags. The top-right section capture alignments of semantically related words, based on relational information extracted from various linguistic resources, such as WordNet and NomBank. And the bottom section capture syntactic edits. Note that multiple edit operations can fire at the same edit position if conditions are matched (e.g., we can choose to delete if there are more words to edit in *txt*, or to insert if there are more words to edit in *hyp*).

5 Features

One of the most distinctive advantages of our model compared to previous tree-edit based models is the ability to include a wide range of non-independent, rich linguistic features. The features we employed can be broken down into two categories. The first category is *zero-order* features that model the current edit step. They consist of a conditioning property of the current edit, and the current state in the FSM. The second category is *first-order* features that capture state transitions, by concatenating the current FSM state with the previous FSM state. One simple form of zero-order feature is the current FSM state itself. The FSM states already carry a lot of information about the current edits. Conditioning properties are used to further describe the current edit. They are often more fine-grained and complex (e.g.,

Surface edits		Semantic edits	
{I,D,S}-{POS}	insert/delete/substitute words of a POS type, where POS is *noun*, *verb* or *proper noun*	S-SYNONYM	substitute two words that are synonyms
		S-HYPERNYM	substitute two words that are hypernyms
{I,D,S}-NE	insert/delete/substitute named-entity words	S-ANTONYM	substitute two words that are antonyms
{I,D,S}-LIKE	insert/delete/substitute words that expresses likelihood, e.g., *maybe*, *possibly*	S-ACRONYM	substitute two words in which one is an acronym of the other
{I,D,S}-MODAL	insert/delete/substitute modal verbs, e.g., *can*, *could*, *may*	S-NOMBANK	substitute two words that are related according to NomBank
S-{SAME/DIFF}	the words being substituted are the same or different	S-NUM-0,1	substitute two words that are both numerical values, and 1 if they match, 0 if they mismatch
Syntactic edits			
{I,D,S}-ROOT	insert/delete/substitute root of the trees		
{I,D,S}-{REL}	insert/delete/substitute a tree node of grammatical relation type, where REL is either SUB, OBJ, VC or PRD		

Table 1: List of edit operations. I for INSERT, D for DELETE, and S for SUBSTITUTE.

syntactic-matching conditions listed below). To give an example, in Figure 1, the second edit operation in the example sequence is S-NE. A matching condition feature that fires with this state could be *substitute_NE_type_PERSON*, which tells us exactly what type of named-entity is being substituted.

It is notable that in designing edit operations and features, there is a continuum of choice in terms of how much information to be encoded as features versus edit operations. To better illustrate the trade-off, consider the two extreme cases of this continuum. At one extreme, we can design a system where there are only three basic edit operations, and all extra information in our current set of edit operations can be encoded as features. For example, in this case edit operation S-NE would become S with feature *substitute_NE*. The other extreme is to encode every zero-order feature as a separate edit operation. The amount of information encoded in the zero-order features and edit operations is the same in both cases, but the difference lies in first-order features and efficiency. When encoding more information as edit operations (and thus more states in FSM), first-order features become much more expressive; whereas when encoding more information as features, computation becomes cheaper as the number of possible state transition sequences is reduced. In our experiments, we aim to keep a minimal set of edit operations that are meaningful but not overly verbose, and encode additional information as features. Each feature is a binary feature initialized with weight 0.

Due to space limitation, we list the most important zero-order features. Many of these features are inspired by MacCartney et al. (2006) and Snow et al. (2006), but not as sophisticated.

Word matching features. These features detect if a text word and a hypothesis word match the following conditions:

1. have the same lemma
2. one is a phrase and contains the other word
3. are multi-word phrases and parts match
4. have the same/different named-entity type(s) + the named-entity type(s)

Tree structure features. These features try to capture syntactic matching/mismatching information from the labeled dependency parse trees. 1. whether the roots of the two trees are aligned
2. parent-child pair match
3. (2.) and labels also match
4. (2.) and labels mismatch
5. (4.) and detailing the mismatching labels
6. parent+label match, child mismatch
7. child and label match, parents are {hyper/syno/anto}nym
8. looking for specific SUB/OBJ/PRD construct as in Snow et al. (2006).

6 Preprocessing

In all of our experiments, each input pair of text and hypothesis sentence is preprocessed as following: Sentences were first tokenized by the standard Penn TreeBank tokenization script, and then we used MXPOST tagger (Ratnaparkhi, 1996) for part-of-speech (POS) tagging. POS tagged sentences were then parsed by MST-Parser (McDonald et al., 2005) to produce labeled dependency parse trees. The parser was trained

on the entire Penn TreeBank. The last step in the pipeline is named-entity tagging using Stanford NER Tagger (Finkel et al., 2005).

7 RTE Experiments

Given an input text sentence and a hypothesis sentence, the task of RTE is to make predictions about whether or not the hypothesis can be entailed from the text sentence. We use standard evaluation datasets RTE1-3 from the Pascal RTE Challenges (Dagan et al., 2006). For each RTE dataset, we train a tree-edit CRF model on the training portion and evaluate on the testing portion. We report accuracy of classification results, and precision and recall for the true entailment class. There is a balanced positive-negative sample distribution in each dataset, so a random baseline gives 50% classification accuracy. We used RTE1 for feature selection and tuning σ in the L_2 regularizer ($\sigma = 5$ was used). RTE2 and RTE3 were reserved for testing.

Our system is compared with four systems on RTE2 and three other systems on the RTE3 dataset.[1] We chose these systems for comparison because they make use of syntactic dependencies and lexical semantic information. Notably other systems that give state-of-the-art performance on RTE use non-comparable techniques such as theorem-proving and logical induction, and often involve significant manual engineering specifically for RTE, thus do not make meaningful comparison to our model.

For RTE2, Kouylekov and Magnini (2006) experimented with various TED cost functions and found a combination scheme to work the best for RTE. Vanderwende et al. (2006) used syntactic heuristic matching rules with a lexical-similarity back-off model. Nielsen et al. (2006) extracted features from dependency path, and combined them with word-alignment features in a mixture of experts classifier. Zanzotto et al. (2006) proposed a syntactic cross-pair similarity measure for RTE.

For RTE3, Harmeling (2007) took a similar classification-based approach with transformation sequence features. Marsi et al. (2007) described a system using dependency-based paraphrasing techniques for RTE. de Marneffe et al. (2006) described a system where best alignments between the sentence pairs were first found, then classification decisions were made based on these alignments.

[1] Different systems are used for comparison because none of these systems reported performance on both datasets.

RTE2	**Acc.%**	**Prec.%**	**Rec.%**
Vanderwende et al., 2006	60.2	59.0	67.0
K&M, 2006	60.5	58.9	70.0
Nielsen et al., 2006	61.1	59.0	73.3
Zanzotto et al., 2006	63.9	60.8	78.0
Tree-edit CRF	63.0	61.7	68.5
RTE3	**Acc.%**	**Prec.%**	**Rec.%**
Marsi et al., 2007	59.1	-	-
Harmeling, 2007	59.5	-	-
de Marneffe et al., 2006	60.5	61.8	60.2
Tree-edit CRF	61.1	61.3	65.3

Table 2: Results on RTE2 and RTE3 dataset. Results for de Marneffe et al. (2006) were reported by MacCartney and Manning (2008).

Table 2 presents RTE results. Our model performs competitively on both datasets. On RTE2, our model gives second best performance among the methods we compare against, and the difference in accuracy from the best system is quite small (7 out of 800 examples). We observe a larger gap in recall, suggesting our method tends to give higher precision, which is also commonly found in other syntax-based systems (Snow et al., 2006). It is worth noting that Zanzotto et al. (2006) achieved second place in the official RTE2 evaluation. On RTE3, our model outperforms the other syntax-based systems compared. In particular, out system gives the same precision level as the second best system (de Marneffe et al., 2006) without sacrificing as much recall, which is the most common drawback found in syntax-based systems.

8 QA Experiments

A second Tree-edit CRF model was trained for the task of answer selection for Question Answering. In this task, the input pair consists of a short factoid question (e.g., *Who beat Floyd Patterson to take the title away?*) and an answer candidate sentence (e.g., *He saw Ingemar Johansson knock down Floyd Patterson seven times there in winning the heavyweight title.*). The pair is judged positive if the answer candidate sentence correctly answers the question and provides sufficient con-

System	MAP	MRR
Punyakanok et al., 2004	0.4189	0.4939
Cui et al., 2005	0.4350	0.5569
Wang et al., 2007	0.6029	0.6852
H&S, 2010	**0.6091**	0.6917
Tree-edit CRF	0.5951	**0.6951**

Table 3: Results on QA task reported in Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR).

textual support (i.e., does not merely contain the answer key, for example, *"Ingemar Johansson was a world heavyweight champion"* would not be a correct answer). We followed the same experimental setup as Wang et al. (2007) and Heilman and Smith (2010). The training portion of the dataset consists of 5919 manually judged Q/A pairs from previous QA tracks at Text REtrieval Conference (TREC 8–12). There are also 1374 Q/A pairs for development and 1866 Q/A pairs for testing, both from the TREC 3 evaluation. The task is framed as a sentence retrieval task, and thus Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) are reported for the ranked list of most probable answer candidates. We compare out model with four other systems. Wang et al. (2007) proposed a Quasi-synchronous Grammar formulation of the problem which also models alignment as structured latent variables, but in a generative probabilistic model. Their method gives the current state-of-the-art performance on this task. Heilman and Smith (2010) presented a classification-based approach with tree-edit features extracted from a tree kernel. Cui et al. (2005) developed a dependency-tree based information discrepancy measure. Punyakanok et al. (2004) used a generalized Tree-edit Distance method to score mappings between dependency parse trees. All systems were evaluated against the same dataset as the one we used. Results of replicated systems for the last two were reported by Wang et al. (2007), with lexical-semantic augmentation from WordNet.

Results in Table 3 show that our model gives the same level of performance as Wang et al. (2007), with no statistically significant difference ($p > 5$ in sign test). Both systems out-perform the other two earlier systems significantly.

9 Discussion

Our experiments on RTE and QA applications demonstrated that Tree-edit CRF models provide results competitive with previous syntax-based methods. Even though the improvements were quite moderate in some cases, the important point is that our model provides a novel principled framework. It works across different problem domains with minimal domain knowledge and feature engineering, whereas previous methods are only engineered for a particular task and are hard to generalize to new problems.

While the current Tree-edit CRF model can model a large set of linguistic phenomenon and tree-transformations, it has some clear limitations. One of the biggest drawbacks is the lack of support for modeling phrasal re-ordering, which is a very common and important linguistic phenomena. It is not straightforward to implement reordering in the current model because it breaks the word-order constraint which admits tractable forward-backward style dynamic programming. However, this shortcoming can be addressed partially by extending the model to deal with constrained re-ordering per Zhang (1996).

10 Related Work

Tree Edit Distance (TED) have been studied extensively in theoretical and algorithmic research (Klein, 1989; Zhang and Shasha, 1989; Bille, 2005). In recent years we have seen many work on applying TED based methods for NLP-related tasks (Punyakanok et al., 2004; Kouylekov and Magnini, 2006; Harmeling, 2007; Mehdad, 2009). Mehdad (2009) proposed a method based on particle swarm optimization technique to automatically learn the TED cost function. Another work that also developed an interesting approach to stochastic tree edit distance is Bernard et al. (2008), but unfortunately experiments in the paper were limited to digit recognition and tasks on small artificial datasets.

Many different approaches to modeling sentence alignment have been proposed before (Haghighi et al., 2005; MacCartney et al., 2008). Haghighi et al. (2005) treated alignment finding in RTE as a graph matching problem

between sentence parse trees. MacCartney et al. (2008) described a phrase-based alignment model for MT, trained by the Perceptron learning algorithm. A line of work that offers similar treatment of alignment to our model is the Quasi-synchronous Grammar (QG) (Smith and Eisner, 2006; Wang et al., 2007; Das and Smith, 2009). QG models alignments between two parse trees as structured latent variables. The generative story of QG describes one that builds the parse tree of one sentence, loosely conditioned on the parse tree of the other sentence. This formalism prefers but is not confined to tree isomorphism, therefore possesses more model flexibility than synchronous grammars.

The work of McCallum et al. (2005) inspired the discriminative training framework that we used in our experiments. They presented a String Edit Distance model that also learns alignments as hidden structures for simple tasks such as restaurant name matching.

Our work is also closely related to other recent work on learning probabilistic models involving structural latent variables (Clark and Curran, 2004; Petrov et al., 2007; Blunsom et al., 2008; Chang et al., 2010). The Tree-edit CRF model we present here is a new addition to this family of interesting models for discriminative learning with structural latent variables.

11 Conclusion

We described a Tree-edit CRF model for predicting semantic relatedness of pairs of sentences. Our approach generalizes TED in a principled probabilistic model that embeds alignments as structured latent variables. We demonstrate a wide-range of lexical-semantic and syntactic features can be easily incorporated into the model. Discriminatively trained, the Tree-edit CRF led to competitive performance on the task of Recognizing Textual Entailment and answer selection for Question Answering.

References

Bernard, M., L. Boyer, A. Habrard, and M. Sebban. 2008. Learning probabilistic models of tree edit distance. *Pattern Recognition*, 41(8):2611–2629.

Bille, P. 2005. A survey on tree edit distance and related problems. *Theoretical Computer Science*, 337(1-3):217–239.

Blunsom, P., T. Cohn, and M. Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *Proceedings of ACL-HLT*.

Chang, Ming-Wei, Dan Goldwasser, Dan Roth, and Vivek Srikumar. 2010. Discriminative learning over constrained latent representations. In *Proceedings of NAACL-HLT*.

Clark, S. and J. R. Curran. 2004. Parsing the wsj using ccg and log-linear models. In *Proceedings of ACL*.

Cui, Hang, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. 2005. Question answering passage retrieval using dependency relations. In *Proceedings of SIGIR*.

Dagan, I., O. Glickman, and B. Magnini. 2006. The pascal recognising textual entailment challenge. *Machine Learning Challenges, LNCS*, 3944:177–190.

Das, Dipanjan and Noah A. Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of ACL-IJCNLP*.

de Marneffe, M.-C., B. MacCartney, T. Grenager, D. Cer, A. Rafferty, and C. D. Manning. 2006. Learning to distinguish valid textual entailments. In *Proceedings of the second PASCAL Challenges Workshop on RTE*.

de Salvo Braz, R., R. Girju, V. Punyakanok, D. Roth, and M. Sammons. 2005. An inference model for semantic entailment and question-answering. In *Proceedings of AAAI*.

Finkel, J. R., T. Grenager, and C. D. Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of ACL*.

Finkel, J. R., C. D. Manning, and A. Y. Ng. 2006. Solving the problem of cascading errors: Approximate bayesian inference for linguistic annotation pipelines. In *Proceedings of EMNLP*.

Haghighi, A., A. Y. Ng, and C. D. Manning. 2005. Robust textual inference via graph matching. In *Proceedings of EMNLP*.

Harmeling, S. 2007. An extensible probabilistic transformation-based approach to the third recognizing textual entailment challenge. In *Proceedings of ACL PASCAL Workshop on Textual Entailment and Paraphrasing*.

Heilman, M. and N. A. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Proceedings of NAACL-HLT*.

Jijkoun, V. and M. de Rijke. 2005a. Recognizing textual entailment: Is word similarity enough?. In *Machine Learning Challenge Workshop*, volume 3944 of *LNCS*, pages 449–460. Springer.

Jijkoun, V. and M. de Rijke. 2005b. Recognizing textual entailment using lexical similarity. In *Proceedings of the PASCAL Challenges Workshop on RTE*.

Klein, P. N. 1989. Computing the edit-distance between unrooted ordered trees. In *Proceedings of European Symposium on Algorithms*.

Kouylekov, M. and B. Magnini. 2006. Tree edit distance for recognizing textual entailment: Estimating the cost of insertion. In *Proceedings of the second PASCAL Challenges Workshop on RTE*.

Lafferty, J., A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*.

MacCartney, Bill and Christopher D. Manning. 2007. Natural logic for textual inference. In *Proceedings of Workshop on Textual Entailment and Paraphrasing at ACL 2007*.

MacCartney, B. and C. D. Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of COLING*.

MacCartney, B., T. Grenager, M.-C. de Marneffe, D. Cer, and C. D. Manning. 2006. Learning to recognize features of valid textual entailments. In *Proceedings of HLT-NAACL*.

MacCartney, B., M. Galley, and C. D. Manning. 2008. A phrase-based alignment model for natural language inference. In *Proceedings of EMNLP*.

Marsi, E., E. Krahmer, and W. Bosma. 2007. Dependency-based paraphrasing for recognizing textual entailment. In *Proceedings of ACL PASCAL Workshop on Textual Entailment and Paraphrasing*.

McCallum, A., K. Bellare, and F. Pereira. 2005. A conditional random field for discriminatively-trained finite-state string edit distance. In *Proceedings of UAI*.

McDonald, R., K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL*.

Mehdad, Yashar. 2009. Automatic cost estimation for tree edit distance using particle swarm optimization. In *Proceedings of ACL*.

Moldovan, D., C. Clark, S. Harabagiu, and S. Maiorano. 2003. Cogex: A logic prover for question answering. In *Proceedings of HLT-NAACL*.

Nielsen, R. D., W. Ward, and J. H. Martin. 2006. Toward dependency path based entailment. In *Proceedings of the second PASCAL Challenges Workshop on RTE*.

Papineni, K., S. Roukos, T. Ward, and W. Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*.

Petrov, S., A. Pauls, and D. Klein. 2007. Discriminative log-linear grammars with latent variables. In *Proceedings of NIPS*.

Punyakanok, V., D. Roth, and W. Yih. 2004. Mapping dependencies trees: An application to question answering. In *Proceedings of AI-Math*.

Raina, R., A. Y. Ng, , and C. Manning. 2005. Robust textual inference via learning and abductive reasoning. In *Proceedings of AAAI*.

Ratnaparkhi, Adwait. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of EMNLP*.

Smith, D. A. and J. Eisner. 2006. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proceedings of the HLT-NAACL Workshop on Statistical Machine Translation*.

Snow, R., L. Vanderwende, and A. Menezes. 2006. Effectively using syntax for recognizing false entailment. In *Proceedings of HLT-NAACL*.

Vanderwende, L., A. Menezes, and R. Snow. 2006. Microsoft research at rte-2: Syntactic contributions in the entailment task: an implementation. In *Proceedings of the second PASCAL Challenges Workshop on RTE*.

Wang, M., N. A. Smith, and T. Mitamura. 2007. What is the jeopardy model? a quasi-synchronous grammar for question answering. In *Proceedings of EMNLP-CoNLL*.

Zanzotto, F. M., A. Moschitti, M. Pennacchiotti, and M.T. Pazienza. 2006. Learning textual entailment from examples. In *Proceedings of the second PASCAL Challenges Workshop on RTE*.

Zhang, K. and D. Shasha. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal of Computing*, 18.

Zhang, K. 1996. A constrained edit distance between unordered labeled trees. *Algorithmica*, 15(3):205–222.

A Character-Based Joint Model for Chinese Word Segmentation

Kun Wang and **Chengqing Zong**
National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Science
{kunwang,cqzong}@nlpr.ia.ac.cn

Keh-Yih Su
Behavior Design Corporation
Kysu@bdc.com.tw

Abstract

The character-based tagging approach is a dominant technique for Chinese word segmentation, and both discriminative and generative models can be adopted in that framework. However, generative and discriminative character-based approaches are significantly different and complement each other. A simple joint model combining the character-based generative model and the discriminative one is thus proposed in this paper to take advantage of both approaches. Experiments on the Second SIGHAN Bakeoff show that this joint approach achieves 21% relative error reduction over the discriminative model and 14% over the generative one. In addition, closed tests also show that the proposed joint model outperforms all the existing approaches reported in the literature and achieves the best F-score in four out of five corpora.

1 Introduction

Chinese word segmentation (CWS) plays an important role in most Chinese NLP applications such as machine translation, information retrieval and question answering. Many statistical methods for CWS have been proposed in the last two decades, which can be classified as either *word-based* or *character-based*. The word-based approach regards the word as the basic unit, and the desired segmentation result is the best word sequence found by the search process. On the other hand, the character-based approach treats the word segmentation task as a character tagging problem. The final segmentation result is thus indirectly generated according to the tag assigned to each associated character. Since the vocabulary size of possible character-tag-pairs is limited, the character-based models can tolerate *out-of-vocabulary* (OOV) words and have become the dominant technique for CWS in recent years.

On the other hand, statistical approaches can also be classified as either adopting a *generative model* or adopting a *discriminative model*. The generative model learns the joint probability of the given input and its associated label sequence, while the discriminative model learns the posterior probability directly. Generative models often do not perform well because they make strong independence assumptions between features and labels. However, (Toutanova, 2006) shows that generative models can also achieve very similar or better performance than the corresponding discriminative models if they have a structure that avoids unrealistic independence assumptions.

In terms of the above dimensions, methods for CWS can be classified as:

1) The word-based generative model (Gao et al., 2003; Zhang et al., 2003), which is a well-known approach and has been used in many successful applications;

2) The word-based discriminative model (Zhang and Clark, 2007), which generates word candidates with both word and character features and is the only word-based model that adopts the discriminative approach;

3) The character-based discriminative model (Xue, 2003; Peng et al., 2004; Tseng et al., 2005; Jiang et al., 2008), which has become the dominant method as it is robust on OOV words and is capable of handling a range of different features, and it has been adopted in many previous works;

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1173–1181,
Beijing, August 2010

4) The character-based generative model (Wang et al., 2009), which adopts a character-tag-pair-based *n*-gram model and achieves comparable results with the popular character-based discriminative model.

In general, character-based models are much more robust on OOV words than word-based approaches do, as the vocabulary size of characters is a closed set (versus the open set of that of words). Furthermore, among those character-based approaches, the generative model and the discriminative one complement each other in handling *in-vocabulary* (IV) words and OOV words. Therefore, a character-based joint model is proposed to combine them.

This proposed joint approach has achieved good balance between IV word recognition and OOV word identification. The experiments of closed tests on the second SIGHAN Bakeoff (Emerson, 2005) show that the joint model significantly outperforms the baseline models of both generative and discriminative approaches. Moreover, statistical significance tests also show that the joint model is significantly better than all those state-of-the-art systems reported in the literature and achieves the best F-score in four of the five corpora tested.

2 Character-Based Models for CWS

The goal of CWS is to find the corresponding word sequence for a given character sequence. Character-based model is to find out the corresponding tags for given character sequence.

2.1 Character-Based Discriminative Model

The character-based discriminative model (Xue, 2003) treats segmentation as a tagging problem, which assigns a corresponding tag to each character. The model is formulated as:

$$P(t_1^n \mid c_1^n) = \prod_{k=1}^{n} P(t_k \mid t_1^{k-1}, c_1^n) \approx \prod_{k=1}^{n} P(t_k \mid c_{k-2}^{k+2}) \qquad (1)$$

Where t_k is a member of {***B**egin*, ***M**iddle*, ***E**nd*, ***S**ingle*} (abbreviated as B, M, E and S from now on) to indicate the corresponding position of character c_k in its associated word. For example, the word "北京市 (Beijing City)" will be assigned with the corresponding tags as: "北/B (North) 京/M (Capital) 市/E (City)".

Since this tagging approach treats characters as basic units, the vocabulary size of those possible character-tag-pairs is limited. Therefore, this method is robust to OOV words and could possess a high *recall of OOV words* (R_{OOV}). Although the dependency between adjacent tags/labels can be addressed, the dependency between adjacent characters within a word cannot be directly modeled under this framework. Lower *recall of IV words* (R_{IV}) is thus usually accompanied (Wang et al., 2009).

In this work, the character-based discriminative model is implemented by adopting the feature templates given by (Ng and Low, 2004), but excluding those ones that are forbidden by the closed test regulation of SIGHAN (e.g., $Pu(C_0)$: whether C_0 is a punctuation). Those feature templates adopted are listed below:

$(a)\, C_n (n = -2, -1, 0, 1, 2);$

$(b)\, C_n C_{n+1} (n = -2, -1, 0, 1);$

$(c)\, C_{-1} C_1$

For example, when we consider the third character "奥" in the sequence "北京奥运会", template (a) results in the features as following: C_{-2}=北, C_{-1}=京, C_0=奥, C_1=运, C_2=会, and template (b) generates the features as: $C_{-2}C_{-1}$=北京, $C_{-1}C_0$=京奥, C_0C_1=奥运, C_1C_2=运会, and template (c) gives the feature $C_{-1}C_1$=京运.

2.2 Character-Based Generative Model

To incorporate the dependency between adjacent characters in the character-based approach, (Wang et al., 2009) proposes a character-based generative model. In this approach, word w_i is first replaced with its corresponding sequence of [character, tag] (denoted as [*c*, *t*]), where tag is the same as that adopted in the above character-based discriminative model. With this representation, this model can be expressed as:

$$\begin{aligned} &P(w_1^m \mid c_1^n) \equiv P([c,t]_1^n \mid c_1^n) \\ &= P(c_1^n \mid [c,t]_1^n) \times P([c,t]_1^n) \big/ P(c_1^n) \end{aligned} \qquad (2)$$

Since $P(c_1^n \mid [c,t]_1^n) \equiv 1$ and $P(c_1^n)$ is the same for various candidates, only $P([c,t]_1^n)$ should be considered. It can be further simplified with Markov Chain assumption as:

$$P([c,t]_1^n) \approx \prod_{i=1}^{n} P([c,t]_i \mid [c,t]_{i-k}^{i-1}). \qquad (3)$$

Compared with the character-based discriminative model, this generative model keeps the capability to handle OOV words because it also regards the character as basic unit. In addition, the dependency between adjacent

宿	Gold and Discriminative Tag: M					Generative Trigram Tag: E				
Tag probability:	B/0.0333		E/0.2236			M/0.7401		S/0.0030		
Feature / Tag	C_{-2}	C_{-1}	C_0	$\boldsymbol{C_1}$	$\boldsymbol{C_2}$	$C_{-2}C_{-1}$	$C_{-1}C_0$	C_0C_1	C_1C_2	$C_{-1}C_1$
B	-1.4375	0.1572	0.0800	0.2282	0.7709	0.2741	0.0000	0.0000	-0.6718	0.0000
E	1.3558	0.1910	0.7229	**-1.2696**	**-0.5970**	0.0049	0.0921	0.0000	0.8049	0.0000
M	1.1071	-0.5527	-0.3174	**2.9422**	**0.4636**	-0.1708	0.0000	0.0000	-0.9700	0.0000
S	-1.0254	0.2046	-0.4856	-1.9008	-0.6375	0.0000	0.0000	0.0000	0.8368	0.0000
者	Gold and Discriminative Tag: E					Generative Trigram Tag: S				
Tag probability:	B/0.0009		E/0.8138			M/0.0012		S/0.1841		
Feature / Tag	C_{-2}	C_{-1}	$\boldsymbol{C_0}$	$\boldsymbol{C_1}$	C_2	$C_{-2}C_{-1}$	$C_{-1}C_0$	C_0C_1	C_1C_2	$C_{-1}C_1$
B	0.3586	0.4175	0.0000	-0.7207	0.4626	0.0085	0.0000	0.0000	0.0000	0.0000
E	0.3666	0.0687	**4.5381**	**2.8300**	-0.0846	0.0000	0.0000	-1.0279	0.6127	0.0000
M	-0.5657	-0.4330	1.8847	0.0000	-0.0918	0.0000	0.0000	0.0000	0.0000	0.0000
S	-0.1595	-0.0532	**2.7360**	**1.8223**	-0.2862	-0.0024	0.0000	1.0494	0.7113	0.0000

Table 1: The corresponding lambda weight of features for "露宿者" in the sentence "[該] [處] [的] [露宿者] [只] [有] [數] [人]". In the Feature column and Tag row, the value is the corresponding lambda weight for the feature and tag under ME framework. The meanings of those features are explained in Section 2.1.

characters is now directly modeled. This will give sharper preference when the history of assignment is given. Therefore, this approach not only holds robust IV performance but also achieves comparable results with the discriminative model. However, the OOV performance of this approach is still lower than that of the discriminative model (see in Table 5), which would be discussed in the next section.

3 Problems with the Character-Based Generative Model

The character-based generative model can handle the dependency between adjacent characters and thus performs well on IV words. However, this generative trigram model is derived under the second order Markov Chain assumption. Future character context (i.e., C_1 and C_2) is thus not utilized in the model when the tag of the current character (i.e., t_0) is determined. Nevertheless, the future context would help to select the correct tag when the associated trigram has not been observed in the training-set, which is just the case for those OOV words. In contrast, the discriminative one could get help from the future context in this case. The example given in the next paragraph clearly shows the above situation.

At the sentence "該(that) 處(place) 的(of) 露宿者(street sleeper) 只(only) 有(have) 數(some) 人(person) (There are only some street sleepers in that place)" in the CITYU corpus, "露/B宿/M者/E(street sleeper)" is observed to be an OOV word, while "露/B 宿/E(sleep on the street)" is an IV word, where the associated tag of each character is given after the slash symbol. The character-based generative model wrongly splits "露宿者" into two words "露/B 宿/E" and "者/S (person)", as the associated trigram for "露宿者" is not seen in the training set. However, the discriminative model gives the correct result for "宿/M" and the dominant features come from its future context "者" and "只". Similarly, the future context "只" helps to give the correct tag to "者/E". Table 1 gives the corresponding lambda feature weights (under the Maximum Entropy (ME) (Ratnaparkhi, 1998) framework) for "露宿者" in the discriminative model. It shows that in the column of "C_1" below "宿", the lambda value associated with the correct tag "M" is 2.9422, which is the highest value in that column and is far greater than that of the wrong tag "E" (i.e., -1.2696) assigned by the generative model. Which indicates that the future feature "C_1" is the most useful feature for tagging "宿".

The above example shows the character-based generative model fails to handle some OOV words such as "露宿者" because this approach cannot utilize future context when it is indeed required. However, the future context for the generative model scanning from left to right is just its past context when it scans from right to left. It is thus expected that this kind of

errors will be fixed if we let the model scans from both directions, and then combine their results. Unfortunately, it is observed that these two scanning modes share over 90% of their errors. For example, in CITYU corpus, the left-to-right scan generates 1,958 wrong words and the right-to-left scan results 1,947 ones, while 1,795 of them are the same. Similar behavior can also be observed on other corpora.

To find out what are the problems, 10 errors that are similar to "露宿者" are selected to examine. Among those errors, only one of them is fixed, and "露宿者" still cannot be correctly segmented. Having analyzed the scores of the model scanning from both directions, we found that the original scores (from left-to-right scan) at the stages "者" and "宿" indeed get better if the model scans from right-to-left. However, the score at the stage "露" deteriorates because the useful feature "者" (a past non-adjacent character for "露" when scans form right-to-left) still cannot be utilized when the past context "宿者" as a whole is unseen, when the related probabilities are estimated via modified Kneser-Ney smoothing (Chen and Goodman, 1998) technique.

Two scanning modes seem not complementing each other, which is out of our original expectation. However, we found that the character-based generative model and the discriminative one complement each other much more than the two scanning modes do. It is observed that these two approaches share less than 50% of their errors. For example, in CITYU corpus, the generative approach generates 1,958 wrong words and the discriminative one results 2,338 ones, while only 835 of them are the same.

The statistics of the remaining errors resulted from the generative model and the discriminative model is shown in Table 2. As shown in the table, it can be seen that the generative model and the discriminative model complement each other on handling IV words and OOV words (In the "IV Errors" column, the number of "G+D-" is much more than the "G-D+", while the behavior is reversed in the "OOV Errors" column).

IV Errors			OOV Errors		
G+D-	**G-D+**	**G-D-**	**G+D-**	**G-D+**	**G-D-**
12,027	4,723	7,481	2,384	6,139	3,975

Table 2: Statistics for remaining errors of the character-based generative model and the discriminative one on the second SIGHAN Bakeoff ("G+D-" in the "IV Errors" column means that the generative model segments the IV words correctly but the discriminative one gives wrong results. The meanings of other abbreviations are similar with this one.).

4 Proposed Joint Model

Since the performance of both IV words and OOV words are important for real applications, we need to combine the strength from both models. Among various combining methods, log-linear interpolation combination is a simple but effective one (Bishop, 2006). Therefore, the following character-based joint model is proposed, and a parameter α is used to weight the generative model in a cross-validation set.

$$\begin{aligned} Score(t_k) = \alpha \times \log(P([c,t]_k \,|\, [c,t]_{k-2}^{k-1})) \\ + (1-\alpha) \times \log(P(t_k \,|\, c_{k-2}^{k+2})) \end{aligned} \quad (4)$$

Where t_k indicates the corresponding position of character c_k, and $\alpha\ (0.0 \le \alpha \le 1.0)$ is the weight for the generative model. $Score(t_k)$ will be used during searching the best sequence. It can be seen that these two models are integrated naturally as both are character-based.

Generally speaking, if the "G(or D)+" has a strong preference on the desired candidate, but the "D(or G)-" has a weak preference on its top-1 incorrect candidate, then this combining method would correct most "G+D- (also G-D+)" errors. On the other hand, the advantage of combining two models would vanish if the "G(or D)+" has a weak preference while the "D(or G)-" has a strong preference over their top-1 candidates. In our observation, these two models meet this requirement quite well.

5 Weigh Various Features Differently

For a given observation, intuitively each feature should be trained only once under the ME framework and its associated weight will be automatically learned from the training corpus. However, when we repeat the work of (Jiang et al., 2008), which reports to achieve the state-of-art performance in the data-sets that we adopt, it has been found that some features (e.g., C_0) are unnoticeably trained several times in their model (which are implicitly generated from different feature templates used in the paper). For example, the feature C_0 actually

Corpus	Abbrev.	Encoding	Training Size (Words/Type)	Test Size (Words/Type)	OOV Rate
Academia Sinica (Taipei)	AS	Unicode/Big5	5.45M/141K	122K/19K	0.046
City University of Hong Kong	CITYU	Unicode/Big5	1.46M/69K	41K/9K	0.074
Microsoft Research (Beijing)	MSR	Unicode/CP936	2.37M/88K	107K/13K	0.026
Peking University	PKU(ucvt.)	Unicode/CP936	1.1M/55K	104K/13K	0.058
	PKU(cvt.)	Unicode/CP936	1.1M/55K	104K/13K	0.035

Table 3: Corpus statistics for the second SIGHAN Bakeoff

appears twice, which is generated from two different templates C_n (with n=0, generates C_0) and $[C_0C_n]$ (used in (Jiang et al., 2008), with n=0, generates $[C_0C_0]$). The meanings of features are illustrated in Section 2.1. Those repetitive features also include $[C_{-1}C_0]$ and $[C_0C_1]$, which implicitly appear thrice. And it is surprising to discover that its better performance is mainly due to this implicit feature repetition but the authors do not point out this fact. As all the features adopted in (Jiang et al., 2008) possess binary values, if a binary feature is repeated n times, then it should behave like a real-valued feature with its value to be "n", at least in principle. Inspired by the above discovery, accordingly, we convert all the binary-value features into their corresponding real-valued features. After having transformed binary features into their corresponding real-valued ones, the original discriminative model is re-trained under the ME framework.

This new implementation, which would be named as the character-based discriminative-plus model, just weights various features differently before conducting ME training. Afterwards, it is further combined with the generative trigram model, and is called the character-based joint-plus model.

6 Experiments

The corpora provided by the second SIGHAN Bakeoff (Emerson, 2005) were used in our experiments. The statistics of those corpora are shown in Table 3.

Note that the PKU corpus is a little different from others. In the training set, Arabic numbers and English characters are in full-width form occupying two bytes. However, in the testing set, these characters are in half-width form occupying only one byte. Most researchers in the SIGHAN Bakeoff competition performed a conversion before segmentation (Xiong et al., 2009). In this work, we conduct the tests on both unconverted (ucvt.) case and converted (cvt.) case. After the conversion, the OOV rate of converted corpus is obviously lower than that of unconverted corpus.

To fairly compare the proposed approach with previous works, we only conduct *closed tests*[1]. The metrics *Precision* (**P**), *Recall* (**R**), *F-score* (**F**) (F=2PR/(P+R)), *Recall of OOV* ($\mathbf{R_{OOV}}$) and *Recall of IV* ($\mathbf{R_{IV}}$) are used to evaluate the results.

6.1 Character-Based Generative Model and Discriminative Model

As shown in (Wang et al., 2009), the character-based generative trigram model significantly exceeds its related bigram model and performs the same as its 4-gram model. Therefore, SRI Language Modeling Toolkit[2] (Stolcke, 2002) is used to train the trigram model with modified Kneser-Ney smoothing (Chen and Goodman, 1998). Afterwards, a beam search decoder is applied to find out the best sequence.

For the character-based discriminative model, the ME Package[3] given by Zhang Le is used to conduct the experiments. Training was done with Gaussian prior 1.0 and 300, 150 iterations for AS and other corpora respectively. Table 5 gives the segmentation results of both the character-based generative model and the discriminative model. From the results, it can be seen that the generative model achieves comparable results with the discriminative one and they outperform each other on different corpus. However, the generative model exceeds the discriminative one on R_{IV} (0.973 vs. 0.956) but loses on R_{OOV} (0.511 vs. 0.680). It illustrates that they complement each other.

[1] According to the second Sighan Bakeoff regulation, the closed test could only use the training data directly provided. Any other data or information is forbidden, including the knowledge of characters set, punctuation set, etc.

[2] http://www.speech.sri.com/projects/srilm/

[3] http://homepages.inf.ed.ac.uk/lzhang10/maxent_toolkit.html

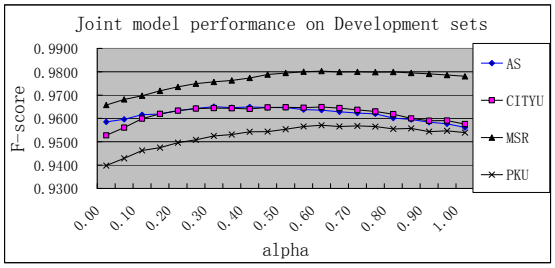


Figure 1: Development sets performance of Character-based joint model.

Corpus	Set	Words	OOV Num	OOV Rate
AS	Development	17,243	445	0.026
	Testing	122,610	5,308/5,311	0.043/0.043
MSR	Development	17,324	355	0.020
	Testing	106,873	2,829/2,833	0.026/0.027
CITYU	Development	12,075	537	0.044
	Testing	40,936	3,028/3,034	0.074/0.074
PKU	Development	13,576	532	0.039
	Testing (ucvt.)	104,372	6,006/6,054	0.058/0.058
	Testing (cvt.)	104,372	3,611/3,661	0.035/0.035

Table 4: Corpus statistics for Development sets and Testing sets. A "/" separates the OOV number (or OOV rate) with respect to the original training sets and the new training sets.

6.2 Character-Based Joint Model

For the character-based joint model, a development set is required to obtain the weight α for its associated generative model. A small portion of each original training corpus is thus extracted as the development set and the remaining data is regarded as the new training-set, which is used to train two new parameter-sets for both generative and discriminative models associated.

The last 2,000, 600, 400, and 300 sentences for AS, MSR, CITYU, and PKU are extracted from the original training corpora as their corresponding development sets. The statistics for new data sets are shown in Table 4. It can be seen that the variation of the OOV rate could be hardly noticed. The F-scores of the joint model, versus different α, evaluated on four development sets are shown in Figure 1. It can be seen that the curves are not sharp but flat near the top, which indicates that the character-based joint model is not sensitive to the α value selected. From those curves, the best suitable α for AS, CITYU, MSR and PKU are found to be 0.30, 0.60, 0.60 and 0.60, respectively. Those alpha values will then be adopted to conduct the experiments on the testing sets.

Corpus	Model	R	P	F	R_{OOV}	R_{IV}
AS	G	0.958	0.938	0.948	0.518	0.978
	D	0.955	0.946	0.951	0.707	0.967
	D-Plus	0.960	0.948	0.954	0.680	0.973
	J	0.962	0.950	**0.956**	0.679	0.975
	J-Plus	0.963	0.949	**0.956**	0.652	0.977
CITYU	G	0.951	0.937	0.944	0.609	0.978
	D	0.941	0.944	0.942	0.708	0.959
	D-Plus	0.951	0.952	0.952	0.720	0.970
	J	0.957	0.951	0.954	0.691	0.979
	J-Plus	0.959	0.952	**0.956**	0.700	0.980
MSR	G	0.974	0.967	0.970	0.561	0.985
	D	0.957	0.962	0.960	0.719	0.964
	D-Plus	0.965	0.967	0.966	0.675	0.973
	J	0.974	0.971	**0.972**	0.659	0.983
	J-Plus	0.975	0.970	**0.972**	0.632	0.984
PKU (ucvt.)	G	0.929	0.933	0.931	0.435	0.959
	D	0.922	0.941	0.932	0.620	0.941
	D-Plus	0.934	0.949	0.941	0.649	0.951
	J	0.935	0.946	0.941	0.561	0.958
	J-Plus	0.937	0.947	**0.942**	0.556	0.960
PKU (cvt.)	G	0.952	0.951	0.952	0.503	0.968
	D	0.940	0.951	0.946	0.685	0.949
	D-Plus	0.949	0.958	0.953	0.674	0.958
	J	0.954	0.958	0.956	0.616	0.966
	J-Plus	0.955	0.958	**0.957**	0.610	0.967
Overall	G	0.953	0.946	0.950	0.511	0.973
	D	0.944	0.950	0.947	0.680	0.956
	D-Plus	0.952	0.955	0.953	0.676	0.965
	J	0.957	0.955	0.956	0.633	0.971
	J-Plus	0.958	0.955	**0.957**	0.621	0.973

Table 5: Segmentation results of various character-based models on the second SIGHAN Bakeoff, the generative trigram model (G), the discriminative model (D), the discriminative-plus model (D-Plus), the joint model (J) and the joint-plus model (J-Plus).

As shown in Table 5, the joint model significantly outperforms both the character-based generative model and the discriminative one in F-score on all the testing corpora. Compared with the generative approach, the joint model increases the overall R_{OOV} from 0.510 to 0.633, with the cost of slightly degrading the overall R_{IV} from 0.973 to 0.971. This shows that the joint model holds the advantage of the generative model on IV words. Compared with the discriminative model, the proposed joint model improves the overall R_{IV} from 0.956 to 0.971, with the cost of degrading the overall R_{OOV} from 0.680 to 0.633. It clearly shows that the joint model achieves a good balance between IV words and OOV words and achieves the best F-scores obtained so far (21% relative error reduction over the discriminative model and 14% over the generative model).

6.3 Weigh Various Features Differently

Inspired by (Jiang et al., 2008), we set the real-value of C_0 to be 2.0, the value of $C_{-1}C_0$ and C_0C_1 to be 3.0, and the values of all other features to be 1.0 for the character-based discriminative-plus model. Although it seems reasonable to weight those closely relevant features more (C_0 should be the most relevant feature for assigning tag t_0), both implementations seem to be equal if their corresponding lambda-values are also updated accordingly. However, Table 5 shows that this new discriminative-plus implementation (D-Plus) significantly outperforms the original one (overall F-score is raised from 0.947 to 0.953) when both of them adopt real-valued features. It is not clear how this change makes the difference.

Similar improvements can be observed with two other ME packages. One anonymous reviewer pointed out that the duplicated features should not make difference if there is no regularization. However, we found that the duplicated features would improve the performance whether we give Gaussian penalty or not.

Afterwards, this new implementation and the generative trigram model are further combined (named as the joint-plus model). Table 5 shows that this joint-plus model also achieves better results compared with the discriminative-plus model, which illustrates that our joint approach is an effective and robust method for CWS. However, compared with the original joint model, the new joint-plus approach does not show much improvement, regardless of the significant improvement made by the discriminative-plus model, as the additional benefit generated by the discriminative-plus model has already covered by the generative approach (Among the 6,965 error words corrected by the discriminative-plus model, 6,292 (90%) of them are covered by the generative model).

7 Statistical Significance Tests

Although Table 5 has shown that the proposed joint (joint-plus) model outperforms all the baselines mentioned above, we want to know if the difference is statistically significant enough to make such a claim. Since there is only one testing set for each training corpus, the bootstrapping technique (Zhang et al., 2004) is adopted to conduct the tests: Giving an

Models		AS	CITYU	MSR	PKU (ucvt.)	PKU (cvt.)
A	B					
G	D	<	~	>	~	>
D-Plus	G	>	>	<	>	>
D-Plus	D	>	>	>	>	>
J	G	>	>	>	>	>
J	D	>	>	>	>	>
J-Plus	G	>	>	>	>	>
J-Plus	D-Plus	>	>	>	~	>
J-Plus	J	~	>	~	>	>

Table 6: Statistical significance test of F-score among various character-based models.

testing-set T_0, additional *M-1* new testing-sets $T_0,\ldots,T_{M-1}$ (each with the same size of T_0) will be generated by repeatedly re-sampling data from T_0. Then, we will have a total of M testing-sets (M=2000 in our experiments).

7.1 Comparisons with Baselines

We then follow (Zhang et al., 2004) to measure the 95% confidence interval for the discrepancy between two models. If the confidence interval does not include the origin point, we then claim that system A is significantly different from system B. Table 6 gives the results of significant tests among various models mentioned above. In this table, ">" means that system A is significantly better than B, where as "<" denotes that system A is significantly worse than B, and "~" indicates that these two systems are not significantly different.

As shown in Table 6, the proposed joint model is significantly better than the two baseline models on all corpora. Similarly, the proposed joint-plus model also significantly outperforms the generative model and the discriminative-plus model on all corpora except on the PKU(ucvt.). The comparison shows that the proposed joint (also joint-plus) model indeed exceeds each of its component models.

7.2 Comparisons with Previous Works

The above comparison mainly shows the superiority of the proposed joint model among those approaches that have been implemented. However, it would be interesting to know if the joint (and joint-plus) model also outperforms those previous state-of-the-art systems.

The systems that performed best for at least one corpus in the second SIGHAN Bakeoff are first selected for comparison. This category includes (Asahara et al., 2005) (denoted as

Asahara05) and (Tseng et al., 2005) [4] (**Tseng05**). (Asahara et al., 2005) achieves the best result in the AS corpus, and (Tseng et al., 2005) performs best in the remaining three corpora. Besides, those systems that are reported to exceed the above two systems are also selected. This category includes (Zhang et al., 2006) (**Zhang06**), (Zhang and Clark, 2007) (**Z&C07**) and (Jiang et al., 2008) (**Jiang08**). They are briefly summarized as follows. (Zhang et al., 2006) is based on sub-word tagging and uses a confidence measure method to combine the sub-word CRF (Lafferty et al., 2001) and rule-based models. (Zhang and Clark, 2007) uses perceptron (Collins, 2002) to generate word candidates with both word and character features. Last, (Jiang et al., 2008)[5] adds repeated features implicitly based on (Ng and Low, 2004). All of the above models, except (Zhang and Clark, 2007), adopt the character-based discriminative approach.

All the results of the systems mentioned above are shown in Table 7. Since the systems are not re-implemented, we cannot generate paired samples from those *M* testing-sets. Instead, we calculate the 95% confidence interval of the joint (also joint-plus) model. Afterwards, those systems can be compared with our proposed models. If the F-score of system B does not fall within the 95% confidence interval of system A (joint or joint-plus), then they are statistically significantly different.

Table 8 gives the results of significant tests for those systems mentioned in this section. It shows that both our joint-plus model and joint model exceed (or are comparable to) almost all the state-of-the-art systems across all corpora, except (Zhang and Clark, 2007) at PKU(ucvt.). In that special case, (Zhang and Clark, 2007)

Participants \ Corpus	AS	CITYU	MSR	PKU (ucvt.)	PKU (cvt.)
Asahara05	0.952	0.941	0.958	N/A	0.941
Tseng05	0.947	0.943	0.964	N/A	0.950
Zhang06	0.951	0.951	0.971	N/A	0.951
Z&C07	0.946	0.951	**0.972**	**0.945**	N/A
Jiang08	0.953	0.948	0.966	0.937	N/A
Our Joint	**0.956**	0.954	**0.972**	0.941	0.956
Our Joint-Plus	**0.956**	**0.956**	**0.972**	0.942	**0.957**

Table 7: Comparisons of F-score with previous state-of-the-art systems.

Systems A	Systems B	AS	CITYU	MSR	PKU (ucvt.)	PKU (cvt.)
J	Asahara05	>	>	>	N/A	>
	Tseng05	>	>	>	N/A	>
	Zhang06	>	~	~	N/A	>
	Z&C07	>	>	~	<	N/A
	Jiang08	>	>	>	>	N/A
J-Plus	Asahara05	>	>	>	N/A	>
	Tseng05	>	>	>	N/A	>
	Zhang06	>	>	~	N/A	>
	Z&C07	>	>	~	<	N/A
	Jiang08	~	>	>	>	N/A

Table 8: Statistical significance test of F-score for previous state-of-the-art systems.

outperforms the joint-plus model by 0.3% on F- score (0.4% for the joint model). However, our joint-plus model exceeds it more over AS and CITYU corpora by 1.0% and 0.5%, respectively (1.0% and 0.3% for the joint model). Thus, it is fair to say that both our joint model and joint-plus model are superior to the state-of-the-art systems reported in the literature.

8 Conclusion

From the error analysis of the character-based generative model and the discriminative one, we found that these two models complement each other on handling IV words and OOV words. To take advantage of these two approaches, a joint model is thus proposed to combine them. Experiments on the Second SIGHAN Bakeoff show that the joint model achieves 21% error reduction over the discriminative model (14% over the generative model). Moreover, closed tests on the second SIGHAN Bakeoff corpora show that this joint model significantly outperforms all the state-of-the-art systems reported in the literature.

Last, it is found that weighting various features differently would give better result. However, further study is required to find out the true reason for this strange but interesting phenomenon.

[4] We are not sure whether (Asahara et al., 2005) and (Tseng et al., 2005) performed a conversion before segmentation in PKU corpus. In this paper, we followed previous works, which cited and compared with them.

[5] The data for (Jiang et al., 2008) given at Table 7 are different from what were reported at their paper. In the communication with the authors, it is found that the script for evaluating performance, provided by the SIGHAN Bakeoff, does not work correctly in their platform. After the problem is fixed, the re-evaluated real performances reported here deteriorate from their original version. Please see the announcement in Jiang's homepage (http://mtgroup.ict.ac.cn/~jiangwenbin/papers/error_correction.pdf).

Acknowledgement

The authors extend sincere thanks to Wenbing Jiang for his helps with our experiments. Also, we thank Behavior Design Corporation for using their Generic-Beam-Search code and show special thanks to Ms. Nanyan Kuo for her helps with the Generic-Beam-Search code.

The research work has been partially funded by the Natural Science Foundation of China under Grant No. 60975053, 90820303 and 60736014, the National Key Technology R&D Program under Grant No. 2006BAH03B02, and also the Hi-Tech Research and Development Program ("863" Program) of China under Grant No. 2006AA010108-4 as well.

References

Masayuki Asahara, Kenta Fukuoka, Ai Azuma, Chooi-Ling Goh, Yotaro Watanabe, Yuji Matsumoto and Takashi Tsuzuki, 2005. Combination of machine learning methods for optimum Chinese word segmentation. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, pages 134–137, Jeju, Korea.

Christopher M. Bishop, 2006. *Pattern recognition and machine learning*. New York: Springer

Stanley F. Chen and Joshua Goodman, 1998. An empirical study of smoothing techniques for language modeling. *Technical Report TR-10-98, Harvard University Center for Research in Computing Technology*.

Michael Collins, 2002. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*, pages 1-8, Philadelphia.

Thomas Emerson, 2005. The second international Chinese word segmentation bakeoff. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, pages 123-133.

Jianfeng Gao, Mu Li and Chang-Ning Huang, 2003. Improved Source-Channel Models for Chinese Word Segmentation. In *Proceedings of ACL*, pages 272-279.

Wenbin Jiang, Liang Huang, Qun Liu and Yajuan Lu, 2008. A Cascaded Linear Model for Joint Chinese Word Segmentation and Part-of-Speech Tagging. In *Proceedings of ACL*, pages 897-904.

John Lafferty, Andrew McCallum and Fernando Pereira, 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of ICML*, pages 282-289.

Hwee Tou Ng and Jin Kiat Low, 2004. Chinese part-of-speech tagging: one-at-a-time or all-at-once? word-based or character-based. In *Proceedings of EMNLP*, pages 277-284.

Fuchun Peng, Fangfang Feng and Andrew McCallum, 2004. Chinese segmentation and new word detection using conditional random fields. In *Proceedings of COLING*, pages 562–568.

Adwait Ratnaparkhi, 1998. Maximum entropy models for natural language ambiguity resolution. University of Pennsylvania.

Andreas Stolcke, 2002. SRILM-an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, pages 311-318.

Kristina Toutanova, 2006. Competitive generative models with structure learning for NLP classification tasks. In *Proceedings of EMNLP*, pages 576-584, Sydney, Australia.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky and Christopher Manning, 2005. A Conditional Random Field Word Segmenter for Sighan Bakeoff 2005. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, pages 168-171.

Kun Wang, Chengqing Zong and Keh-Yih Su, 2009. Which is more suitable for Chinese word segmentation, the generative model or the discriminative one? In *Proceedings of PACLIC*, pages 827-834, Hong Kong, China.

Ying Xiong, Jie Zhu, Hao Huang and Haihua Xu, 2009. Minimum tag error for discriminative training of conditional random fields. *Information Sciences*, 179 (1-2). pages 169-179.

Nianwen Xue, 2003. Chinese Word Segmentation as Character Tagging. *Computational Linguistics and Chinese Language Processing*, 8 (1). pages 29-48.

Huaping Zhang, Hongkui Yu, Deyi Xiong and Qun Liu, 2003. HHMM-based Chinese lexical analyzer ICTCLAS. In *Proceedings of the Second SIGHAN Workshop on Chinese Language Processing*, pages 184–187.

Ruiqiang Zhang, Genichiro Kikui and Eiichiro Sumita, 2006. Subword-based Tagging for Confidence-dependent Chinese Word Segmentation. In *Proceedings of the COLING/ACL*, pages 961-968, Sydney, Australia.

Ying Zhang, Stephan Vogel and Alex Waibel, 2004. Interpreting BLEU/NIST scores: How much improvement do we need to have a better system. In *Proceedings of LREC*, pages 2051–2054.

Yue Zhang and Stephen Clark, 2007. Chinese Segmentation with a Word-Based Perceptron Algorithm. In *Proceedings of ACL*, pages 840-847, Prague, Czech Republic.

Near-synonym Lexical Choice in Latent Semantic Space

Tong Wang
Department of Computer Science
University of Toronto
tong@cs.toronto.edu

Graeme Hirst
Department of Computer Science
University of Toronto
gh@cs.toronto.edu

Abstract

We explore the near-synonym lexical choice problem using a novel representation of near-synonyms and their contexts in the latent semantic space. In contrast to traditional latent semantic analysis (LSA), our model is built on the lexical level of co-occurrence, which has been empirically proven to be effective in providing higher dimensional information on the subtle differences among near-synonyms. By employing supervised learning on the latent features, our system achieves an accuracy of 74.5% in a "fill-in-the-blank" task. The improvement over the current state-of-the-art is statistically significant.

We also formalize the notion of *subtlety* through its relation to semantic space dimensionality. Using this formalization and our learning models, several of our intuitions about subtlety, dimensionality, and context are quantified and empirically tested.

1 Introduction

Lexical choice is the process of selecting content words in language generation. Consciously or not, people encounter the task of lexical choice on a daily basis — when speaking, writing, and perhaps even in inner monologues. Its application also extends to various domains of natural language processing, including Natural Language Generation (NLG, Inkpen and Hirst 2006), writers' assistant systems (Inkpen, 2007), and second language (L2) teaching and learning (Ouyang et al., 2009).

In the context of *near-synonymy*, the process of lexical choice becomes profoundly more complicated. This is partly because of the subtle nuances among near-synonyms, which can arguably differ along an infinite number of dimensions. Each dimension of variation carries differences in style, connotation, or even truth conditions into the discourse in question (Cruse, 1986), all making the seemingly intuitive problem of "choosing the right word for the right context" far from trivial even for native speakers of a language. In a widely-adopted "fill-in-the-blank" task, where the goal was to guess missing words (from a set of near-synonyms) in English sentences, two human judges achieved an accuracy of about 80% (Inkpen, 2007). The current state-of-the-art accuracy for an automated system is 69.9% (Islam and Inkpen, 2010).

When the goal is to make plausible or even elegant lexical choices that best suit the *context*, the representation of that context becomes a key issue. We approach this problem in the *latent semantic space*, where transformed local co-occurrence data is capable of implicitly inducing global knowledge (Landauer and Dumais, 1997). A latent semantic space is constructed by reducing the dimensionality of co-occurring linguistic units — typically words and documents as in *Latent Semantic Analysis* (LSA). We refer to this *level of association* (LoA) as *document LoA* hereafter. Although document LoA can benefit topical level classification (e.g., as in document retrieval, Deerwester et al. 1990), it is not necessarily suitable for lexical-level tasks which might require information on a more fine-grained level (Edmonds and Hirst, 2002). Our experimental results show

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1182–1190,
Beijing, August 2010

noticeable improvement when the co-occurrence matrix is built on a lexical LoA between words within a given context window.

One intuitive explanation for this improvement is that the lexical-level co-occurrence might have helped recover the high-dimensional subtle nuances between near-synonyms. This conjecture is, however, as imprecise as it is intuitive. The notion of *subtlety* has mostly been used qualitatively in the literature to describe the level of difficulty involved in near-synonym lexical choice. Hence, we endeavor to formalize the concept of subtlety computationally by using our observations regarding the relationship between "subtle" concepts and their lexical co-occurrence patterns.

We introduce related work on near-synonymy, lexical choice, and latent semantic space models in the next section. Section 3 elaborates on lexical and contextual representations in latent semantic space. In Section 4, we formulate near-synonym lexical choice as a learning problem and report our system performance. Section 5 formalizes the notion of subtlety and its relation to dimensionality and context. Conclusions and future work are presented in Section 6.

2 Related Work

2.1 Near-Synonymy and Nuances

Near-synonymy is a concept better explained by intuition than by definition — which it does not seem to have in the existing literature. We thus borrow Table 1 from Edmonds and Hirst (2002) to illustrate some basic ideas about near-synonymy. Cruse (1986) compared the notion of *plesionymy* to cognitive synonymy in terms of mutual entailment and semantic traits, which, to the best of our knowledge, is possibly the closest to a textbook account of near-synonymy.

There has been a substantial amount of interest in characterizing the nuances between near-synonyms for a computation-friendly representation of near-synonymy. DiMarco et al. (1993) discovered 38 dimensions for differentiating near-synonyms from dictionary usage notes and categorized them into semantic and stylistic variations. Stede (1993) focused on the latter and further decomposed them into seven scalable sub-categories. By organizing near-synonym variations into a tree structure, Inkpen and Hirst (2006) combined stylistic and attitudinal variation into one class parallel to denotational differences. They also incorporated this knowledge of near-synonyms into a knowledge base and demonstrated its application in an NLG system.

Table 1: Examples of near-synonyms and dimension of variations (Edmonds and Hirst, 2002).

Types of variation	***Examples***
Continuous, intermittent	seep:drip
Emphasis	enemy:foe
Denotational, indirect	error:mistake
Denotational, fuzzy	woods:forest
Stylistic, formality	pissed:drunk:inebriated
Stylistic, force	ruin:annihilate
Expressed attitude	skinny:thin:slim:slender
Emotive	daddy:dad:father
Collocational	task:job
Selectional	pass away:die
Sub-categorization	give:donate

2.2 Lexical Choice Evaluation

Due to their symbolic nature, many of the early studies were only able to provide "demo runs" in NLG systems rather than any empirical evaluation. The study of near-synonym lexical choice had remained largely qualitative until a "fill-in-the-blank" (FITB) task was introduced by Edmonds (1997). The task is based on sentences collected from the 1987 *Wall Street Journal* (WSJ) that contain any of a given set of near-synonyms. Each occurrence of the near-synonyms is removed from the sentence to create a "lexical gap", and the goal is to guess which one of the near-synonyms is the missing word. Presuming that the 1987 WSJ authors have made high-quality lexical choices, the FITB test provides a fairly objective benchmark for empirical evaluation for near-synonym lexical choice. The same idea can be applied to virtually any corpus to provide a fair amount of gold-standard data at relatively low cost for lexical choice evaluation.

The FITB task has since been frequently adopted for evaluating the quality of lexical choice systems on a standard dataset of seven near-synonym sets (as shown in Table 2). Edmonds

(1997) constructed a second-order lexical co-occurrence network on a training corpus (the 1989 WSJ). He measured the *word-word distance* using *t-score* inversely weighted by both distance and order of co-occurrence in the network. For a sentence in the test data (generated from the 1987 WSJ), the candidate near-synonym minimizing the sum of its distance from all other words in the sentence (*word-context distance*) was considered the correct answer. Average accuracy on the standard seven near-synonym sets was 55.7%.

Inkpen (2007) modeled word-word distance using *Pointwise Mutual Information* (PMI) approximated by word counts from querying the *Waterloo Multitext System* (Clarke et al., 1998). Word-context distance was the sum of PMI scores between a candidate and its neighboring words within a window-size of 10. An unsupervised model using word-context distance directly achieved an average accuracy of 66.0%, while a supervised method with lexical features added to the word-context distance further increased the accuracy to 69.2%.

Islam and Inkpen (2010) developed a system which completed a test sentence with possible candidates one at a time. The candidate generating the most probable sentence (measured by a 5-gram language model) was proposed as the correct answer. N-gram counts were collected from *Google Web1T Corpus* and smoothed with *missing counts*, yielding an average accuracy of 69.9%.

2.3 Lexical Choice Outside the Near-synonymy Domain

The problem of lexical choice also comes in many flavors outside the near-synonymy domain. Reiter and Sripada (2002) attributed the variation in lexical choice to cognitive and vocabulary differences among individuals. In their meteorology domain data, for example, the term *by evening* was interpreted as *before 00:00* by some forecasters but *before 18:00* by others. They claimed that NLG systems might have to include redundancy in their output to tolerate cognitive differences among individuals.

2.4 Latent Semantic Space Models and LoA

LSA has been widely applied in various fields since its introduction by Landauer and Dumais (1997). In their study, LSA was conducted on *document* LoA on encyclopedic articles and the latent space vectors were used for solving TOEFL synonym questions. Rapp (2008) used LSA on *lexical* LoA for the same task and achieved 92.50% in accuracy in contrast to 64.38% given by Landauer and Dumais (1997). This work confirmed our early postulation that document LoA might not be tailored for lexical level tasks, which might require lower LoAs for more fine-grained co-occurrence knowledge. Note, however, that confounding factors might also have led to the difference in performance, since the two studies used different weighting schemes and different corpora for the co-occurrence model[1]. In Section 3.2 we will compare models on the two LoAs in a more controlled setting to show their difference in the lexical choice task.

3 Representing Words and Contexts in Latent Semantic Space

We first formalize the FITB task to facilitate later discussions. A test sentence $t = \{w_1, \ldots, w_{j-1}, s_i, w_{j+1}, \ldots, w_m\}$ contains a near-synonym s_i which belongs to a set of synonyms $S = \{s_1, \ldots, s_n\}, 1 \leq i \leq n$. A FITB test case is created by removing s_i from t, and the *context* (the incomplete sentence) $c = t - \{s_i\}$ is presented to subjects with a set of possible choices S to guess which of the near-synonyms in S is the missing word.

3.1 Constructing the Latent Space Representation

The first step in LSA is to build a *co-occurrence matrix M* between words and documents, which is further decomposed by *Singular Value Decomposition* (SVD) according to the following equation:

$$M_{v\times d} = U_{v\times k}\Sigma_{k\times k}V^T_{k\times d}$$

[1]The former used *Groliers Academic American Encyclopedia* with weights divided by word entropy, while the latter used the *British National Corpus* with weights multiplied by word entropy.

Here, subscripts denote matrix dimensions, U, Σ, and V together create a decomposition of M, v and d are the number of word types and documents, respectively, and k is the number of dimensions for the latent semantic space. A word w is represented by the row in U corresponding to the row for w in M. For a context c, we construct a vector $\mathbf{c}$ of length v with zeros and ones, each corresponding to the presence or absence of a word w_i with respect to c, i.e.,

$$\mathbf{c}_i = \begin{cases} 1 & \text{if } w_i \in c \\ 0 & \text{otherwise} \end{cases}$$

We then take this *lexical space* vector $\mathbf{c}_{v\times 1}$ as a *pseudo-document* and transform it into a *latent semantic space* vector $\hat{c}$:

$$\hat{\mathbf{c}} = \Sigma^{-1} U^T \mathbf{c} \quad (1)$$

An important observation is that this representation is equivalent to a *weighted centroid* of the context word vectors: when $\mathbf{c}$ is multiplied by $\Sigma^{-1}U^T$ in Equation (1), the product is essentially a weighted sum of the rows in U corresponding to the context words. Consequently, simple modifications on the weighting can yield other interesting representations of context. Consider, for example, the weighting vector $\mathbf{w}_{k\times 1} = (\sigma_1, \cdots, \sigma_k)^T$ with

$$\sigma_i = \frac{1}{|2(p_{\text{gap}} - i) - 1|}$$

where p_{gap} is the position of the "gap" in the test sentence. Multiplying $\mathbf{w}$ before Σ^{-1} in Equation (1) is equivalent to giving the centroid gradient-decaying weights with respect to the distance between a context word and the near-synonym. This is a form of a *Hyperspace Analogue to Language* (HAL) model, which is sensitive to word order, in contrast to a *bag-of-words* model.

3.2 Dimensionality and Level of Association

The number of dimensions k is an important choice to make in latent semantic space models. Due to the lack of any principled guideline for doing otherwise, we conducted a brute force grid search for a proper k value for each LoA, on the basis of the performance of the unsupervised model (Section 4.1 below).

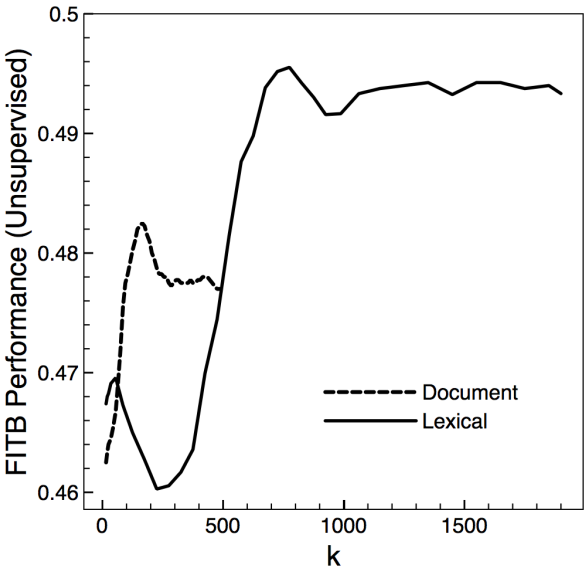


Figure 1: FITB Performance on different LoAs as a function of the latent space dimensionality.

In Figure 1, performance on FITB using this unsupervised model is plotted against k for document and lexical LoAs. Document LoA is very limited in the available number of dimensions[2]; higher dimensional knowledge is simply unavailable from this level of co-occurrence. In contrast, lexical LoA stands out around $k = 550$ and peaks around $k = 700$. Although the advantage of lexical LoA in the unsupervised setting is not significant, later we show that lexical LoA nonetheless makes higher-dimensional information available for other learning methods.

Note that the scale on the y-axis is stretched to magnify the trends. On a zero-to-one scale, the performance of these unsupervised methods is almost indistinguishable, indicating that the unsupervised model is not capable of using the high-dimensional information made available by lexical LoA. We will elaborate on this point in Section 5.2.

[2]The dimensions for document and lexical LoAs on our development corpus are $55,938 \times 500$ and $55,938 \times 55,938$, respectively. The difference is measured between $v \times d$ and $v \times v$ (Section 3.1).

4 Learning in the Latent Semantic Space

4.1 Unsupervised Vector Space Model

When measuring distance between vectors, LSA usually adopts regular vector space model distance functions such as *cosine similarity*. With the context being a centroid of words (Section 3.1), the FITB task then becomes a *k-nearest neighbor* problem in the latent space with $k = 1$ to choose the best near-synonym for the context:

$$s^* = \underset{s_i}{\operatorname{argmax}} \cos(U_{\text{rowId}(\mathbf{v}(s_i),M)}, \hat{\mathbf{c}})$$

where $\mathbf{v}(s_i)$ is the corresponding row for near-synonym s_i in M, and $\text{rowId}(\mathbf{v}, M)$ gives the row number of a vector $\mathbf{v}$ in a matrix M containing $\mathbf{v}$ as a row.

In a model with a cosine similarity distance function, it is detrimental to use Σ^{-1} to weight the context centroid $\hat{\mathbf{c}}$. This is because elements in Σ are the singular values of the co-occurrence matrix along its diagonal, and the amplitude of a singular value (intuitively) corresponds to the significance of a dimension in the latent space; when the inverted matrix is used to weight the centroid, it will "misrepresent" the context by giving more weight to less-significantly co-occurring dimensions and thus sabotage performance. We thus use Σ instead of Σ^{-1} in our experiments. As shown in Figure 1, the best unsupervised performance on the standard FITB dataset is 49.6%, achieved on lexical LoA at $k = 800$.

4.2 Supervised Learning on the Latent Semantic Space Features

In traditional latent space models, the latent space vectors have almost invariantly been used in the unsupervised setting discussed above. Although the number of dimensions has been reduced in the latent semantic space, the inter-relations between the high-dimension data points may still be complex and non-linear; such problems lend themselves naturally to supervised learning.

We therefore formulate the near-synonym lexical choice problem as a supervised classification problem with latent semantic space features. For a test sentence in the FITB task, for example, the context is represented as a latent semantic space vector as discussed in Section 3.1, which is then paired with the correct answer (the near-synonym removed from the sentence) to form one training case.

We choose *Support Vector Machines* (SVMs) as our learning algorithm for their widely acclaimed classification performance on many tasks as well as their noticeably better performance on the lexical choice task in our pilot study. Table 2 lists the supervised model performance on the FITB task together with results reported by other related studies. The model is trained on the 1989 WSJ and tested on the 1987 WSJ to ensure maximal comparability with other results. The optimal k value is 415. Context window size[3] around the gap in a test sentence also affects the model performance. In addition to using the words in the original sentence, we also experiment with enlarging the context window to neighboring sentences and shrinking it to a window frame of n words on each side of the gap. Interestingly, when making the lexical choice, the model tends to favor more-local information — a window frame of size 5 gives the best accuracy of 74.5% on the test. Based on *binomial exact test*[4] with a 95% confidence interval, our result outperforms the current state-of-the-art with statistical significance.

5 Formalizing Subtlety in the Latent Semantic Space

In this section, we formalize the notion of subtlety through its relation to dimensionality, and use the formalization to provide empirical support for some of the common intuitions about subtlety and its complexity with respect to dimensionality and size of context.

5.1 Characterizing Subtlety Using Collocating Differentiator of Subtlety

In language generation, subtlety can be viewed as a subordinate semantic trait in a linguistic realiza-

[3] Note that the *context window* in this paragraph is implemented on FITB test cases, which is different from the context size we compare in Section 5.3 for building co-occurrence matrix.

[4] The binomial nature of the outcome of an FITB test case (right or wrong) makes *binomial exact test* a more suitable significance test than the *t-test* used by Inkpen (2007).

Table 2: Supervised performance on the seven standard near-synonym sets in the FITB task. *95% Confidence* based on *Binomial Exact Test*.

Near-synonyms	Co-occur. network (Edmonds, 1997)	SVMs & PMI (Inkpen, 2007)	5-gram language model (Islam and Inkpen, 2010)	SVMs on latent vectors (Section 4.2)
difficult, hard, tough	47.9%	57.3%	**63.2**%	61.7%
error, mistake, oversight	48.9%	70.8%	78.7%	**82.5**%
job, task, duty	68.9%	**86.7**%	78.2%	82.4%
responsibility, burden, obligation, commitment	45.3%	66.7%	**72.2**%	63.5%
material, stuff, substance	64.6%	71.0%	70.4%	**78.5**%
give, provide, offer	48.6%	56.1%	55.8%	**75.4**%
settle, resolve	65.9%	75.8%	70.8%	**77.9**%
Average	55.7%	69.2%	69.9%	**74.5**%
Data size	29,835	31,116	31,116	30,300
95% confidence	55.1–56.3%	68.7–69.7%	69.3–70.4%	74.0–75.0%

tion of an intention[5]. A key observation regarding subtlety is that it is non-trivial to *characterize* subtle differences between two linguistic units by their collocating linguistic units. More interestingly, the difficulty in such characterization can be approximated by the difficulty in finding a third linguistic unit satisfying the following constraints:

1. The unit must collocate closely with at least one of the two linguistic units under differentiation;

2. The unit must be characteristic of the difference between the pair.

Such approximation is meaningful in that it transforms the abstract *characterization* into a concrete task of finding this third linguistic unit. For example, suppose we want to find out whether the difference between *glass* and *mug* is subtle. The approximation boils the answer down to the difficulty of finding a third word satisfying the two constraints, and we may immediately conclude that the difference between the pair is *not* subtle since it is relatively easy to find *wine* as the qualifying third word, which 1) collocates closely with *glass* and 2) characterizes the difference between the pair by instantiating one of their major differences — the purpose of use. The same reasoning applies to concluding *non-subtlety* for word pairs such as *pen* and *pencil* with *sharpener*, *weather* and *climate* with *forecast*, *watch* and *clock* with *wrist*, etc.

In contrast, for the pair *forest* and *woods*, it might be easy to find words satisfying one but not both constraints. Consequently, the lack of such qualifying words — or at least the relative difficulty for finding one — makes the difference between this pair more subtle than in the previous examples.

We call a linguistic unit satisfying both constraints a *collocating differentiator of subtlety* (CDS). Notably, the second constraint puts an important difference between CDSs and the conventional sense of collocation. On the lexical level, CDSs are not merely words that collocate more with one word in a pair than with the other; they have to be *characteristic of the differences* between the pair. In the example of *forest* and *woods*, one can easily find a word exclusively collocating with one but not the other — such as *national forest* but not **national woods*; however, unlike the CDSs in the previous examples, the word *national* does not characterize any of the differences between the pair in size, primitiveness,

[5]The same principle applies when we replace "generation" with "understanding" and "an intention" with "a cognition".

proximity to civilization, or wildness (Edmonds and Hirst, 2002), and consequently fails to satisfy the second constraint.

5.2 Relating Subtlety to Latent Space Dimensionality[6]

As mentioned in Section 4.1, elements of a latent space vector are in descending order in terms of co-occurrence significance, i.e., the information within the first few dimensions is obtained from more closely collocating linguistic units. From the two constraints in the previous section, it follows that it should be relatively easier to find a CDS for words that can be well distinguished in a lower-dimensional sub-space of the latent semantic space, and the difference among such words should *not* be considered subtle.

We thus claim that co-occurrence-based information capable of characterizing subtle differences must then reside in higher dimensions in the latent space vectors. Furthermore, our intuition on the complexity of subtlety can also be empirically tested by comparing the performance of supervised and unsupervised models at different k values. One of the differences between the two types of models is that supervised models are better at unraveling the convoluted inter-relations between high-dimensional data points. Under this assumption, if we hypothesize that subtlety is a certain form of complex, high-dimensional relation between semantic elements, then the difference in performance between the supervised and unsupervised model should increase as the former recovers subtle information in higher dimensions.

As shown in Figure 2, performance of both models is positively correlated to the number of dimensions in the latent semantic space (with correlation coefficient $\rho = 0.95$ for supervised model and $\rho = 0.81$ for unsupervised model). This suggests that the lexical choice process is indeed "picking up" implicit information about subtlety in the higher dimensions of the latent vectors. Meanwhile, the difference between the performance of the two models correlates strongly to k with $\rho = 0.95$. Significance tests on the "difference of *difference*"[7] between their performances further reveal increasing difference in growth rate of their performance. Significance is witnessed in both the F-test and the paired t-test,[8] indicating that the subtlety-related information in the higher dimensions exhibits complex clustering patterns that are better recognized by SVMs but beyond the capability of the KNN model.

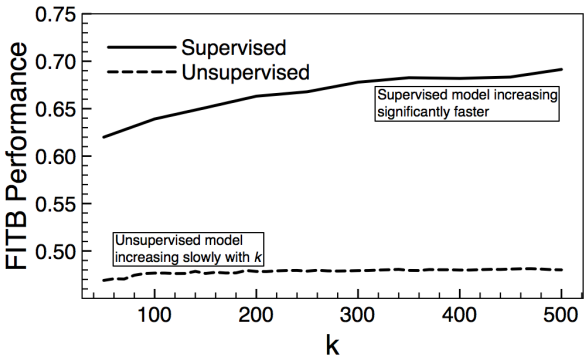


Figure 2: Supervised performance increasing further from unsupervised performance in higher dimensions.

5.3 Subtlety and the Level of Context

Our previous models on lexical LoA associated words within the same sentence to build the co-occurrence matrix. Lexical LoA also allows us to associate words that co-occur in different *levels of context* (LoC) such as paragraphs or documents. This gives an approximate measurement of how much context a lexical LoA model uses for word co-occurrence. Intuitively, by looking at more context, higher LoC models should be better at differentiating more subtle differences.

We compare the performance of models with different LoCs in Figure 3. The sentence LoC model constantly out-performs the paragraph LoC model after $k = 500$, indicating that, by *inter-model comparison*, larger LoC models do not necessarily perform better on higher dimensions. However, there is a noticeable difference in the optimal dimensionality for the model performances. Sentence LoC performance peaks around

[6] In order to keep the test data (1987 *WSJ*) unseen before producing the results in Table 2, models in this section were trained on *The Brown Corpus* and tested on 1988–89 *WSJ*.

[7] The italicized *difference* is used in its mathematical sense as the discrete counterpart of *derivative*.

[8] F-test: $f(1,16) = 9.13, p < 0.01$. Paired t-test: $t(8) = 4.16$ with two-tailed $p = 0.0031$. Both conducted on 10 data points at $k = 50$ to 500 with a step of 50.

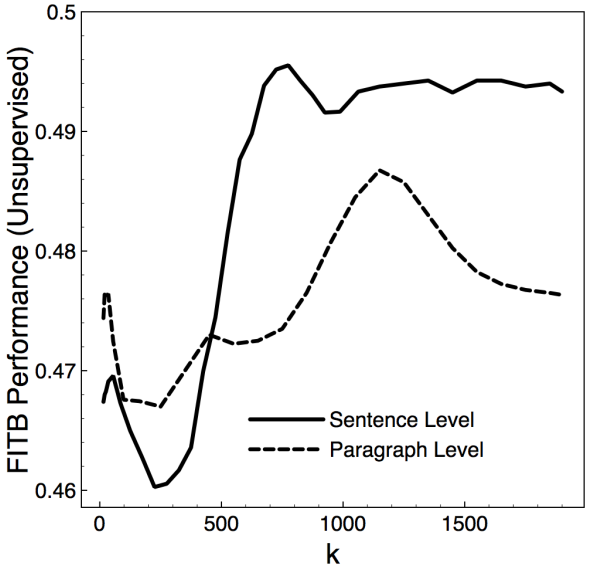


Figure 3: LoC in correlation to latent space dimensionality for optimal model performance.

$k = 700$ — much lower than that of paragraph LoC which is around $k = 1,100$. Such difference may suggest that, by *intra-model comparison*, each model may have its own "comfort zone" for the degree of subtlety it differentiates; models on larger LoC are better at differentiating between more subtle nuances, which is in accordance with our intuition.

One possible explanation for sentence LoC models outperforming paragraph LoC models is that, although the high-dimensional elements are weighed down by Σ due to their insignificance in the latent space, their contribution to the output of distance function is larger in paragraph LoC models because the vectors are much *denser* than that in the sentence LoC model; since the unsupervised method is incapable of recognizing the clustering patterns well in high-dimensional space, the "amplified" subtlety information is eventually taken as noise by the KNN model. An interesting extension to this discussion is to see whether a *supervised* model can consistently perform better on higher LoC in all dimensions.

6 Conclusions and Future Work

We propose a latent semantic space representation of near-synonyms and their contexts, which allows a thorough investigation of several aspects of the near-synonym lexical choice problem. By employing supervised learning on the latent space features, we achieve an accuracy of 74.5% on the "fill-in-the-blank" task, outperforming the current state-of-the-art with statistical significance.

In addition, we formalize the notion of *subtlety* by relating it to the dimensionality of the latent semantic space. Our empirical analysis suggests that subtle differences between near-synonyms reside in higher dimensions in the latent semantic space in complex clustering patterns, and that the degree of subtlety correlates to the level of context for co-occurrence. Both conclusions are consistent with our intuition.

As future work, we will make better use of the easy customization of the context representation to compare HAL and other models with *bag-of-words* models. The correlation between subtlety and dimensionality may lead to many interesting tasks, such as measuring the degree of subtlety for individual near-synonyms or near-synonym sets. With regard to context representation, it is also intriguing to explore other dimensionality reduction methods (such as *Locality Sensitive Hashing* or *Random Indexing*) and to compare them to the SVD-based model.

Acknowledgment

This study is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC). We greatly appreciate the valuable suggestions and feedback from all of our anonymous reviewers and from our colleagues Julian Brooke, Frank Rudzicz, and George Dahl.

References

Charles L. A. Clarke, Gordon Cormack, and Christopher Palmer. An overview of MultiText. *ACM SIGIR Forum*, 32(2):14–15, 1998.

D. A. Cruse. *Lexical Semantics*. Cambridge University Press, 1986.

Scott Deerwester, Susan Dumais, George Furnas, Thomas Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

Chrysanne DiMarco, Graeme Hirst, and Manfred Stede. The semantic and stylistic differentiation of synonyms and near-synonyms. *AAAI Spring Symposium on Building Lexicons for Machine Translation*, pages 114–121, 1993.

Philip Edmonds. Choosing the word most typical in context using a lexical co-occurrence network. In *Proceedings of the 35th annual meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 507–509, 1997.

Philip Edmonds and Graeme Hirst. Near-synonymy and lexical choice. *Computational Linguistics*, 28(2):105–144, 2002.

Diana Inkpen. A statistical model for near-synonym choice. *ACM Transactions on Speech and Language Processing*, 4(1):1–17, 2007.

Diana Inkpen and Graeme Hirst. Building and using a lexical knowledge-base of near-synonym differences. *Computational Linguistics*, 32(2): 223–262, 2006.

Aminul Islam and Diana Inkpen. Near-synonym choice using a 5-gram language model. *Research in Computing Sciences*, 46:41–52, 2010.

Thomas Landauer and Susan Dumais. A solution to Plato's problem: the latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211–240, 1997.

Shixiao Ouyang, Helena Hong Gao, and Soo Ngee Koh. Developing a computer-facilitated tool for acquiring near-synonyms in Chinese and English. In *Proceedings of the Eighth International Conference on Computational Semantics*, pages 316–319, 2009.

Reinhard Rapp. The automatic generation of thesauri of related words for English, French, German, and Russian. *International Journal of Speech Technology*, 11(3):147–156, 2008.

Ehud Reiter and Somayajulu Sripada. Human variation and lexical choice. *Computational Linguistics*, 28(4):545–553, 2002.

Manfred Stede. Lexical choice criteria in language generation. In *Proceedings of the sixth conference of the European Chapter of the Association for Computational Linguistics*, pages 454–459, 1993.

Disambiguating Dynamic Sentiment Ambiguous Adjectives

Yunfang Wu
Key Laboratory of Computational Linguistics (Peking University), Ministry of Education of China
wuyf@pku.edu.cn

Miaomiao Wen*
Department of Electrical Engineering and Information Systems, University of Tokyo
wenmiaomiao98@gmail.com

Abstract

Dynamic sentiment ambiguous adjectives (DSAAs) like "large, small, high, low" pose a challenging task on sentiment analysis. This paper proposes a knowledge-based method to automatically determine the semantic orientation of DSAAs within context. The task is reduced to sentiment classification of target nouns, which we refer to sentiment expectation instead of semantic orientation widely used in previous researches. We mine the Web using lexico-syntactic patterns to infer sentiment expectation of nouns, and then exploit character-sentiment model to reduce noises caused by the Web data. At sentence level, our method achieves promising result with an f-score of 78.52% that is substantially better than baselines. At document level, our method outperforms previous work in sentiment classification of product reviews.

1 Introduction

In recent years, sentiment analysis has attracted considerable attention in the NLP community. It is the task of mining positive and negative opinions from natural language, which can be applied to many research fields. Previous work on this problem falls into three groups: opinion mining of documents, sentiment classification of sentences and polarity prediction of words. Sentiment analysis both at document and sentence level rely heavily on word level.

The most frequently explored task at the word level is to determine the polarity of words, in which most work centers on assigning a prior polarity to words or word senses in the lexicon out of context. However, for some words, the polarity varies strongly with context, making it hard to attach each to a fixed sentiment category in the lexicon. For example, the word "low" has a positive orientation in "low cost" but a negative orientation in "low salary". We call these words like "low" dynamic sentiment ambiguous adjectives (DSAAs). Turney and Littman (2003) claim that DSAAs cannot be avoided in a real-world application. But unfortunately, DSAAs are discarded by most research concerning sentiment analysis.

In this paper, we are devoted to the challenging task of disambiguating DSAAs. The task is to automatically determine the semantic orientation (SO) of DSAAs within context. We limit our work to 14 frequently used adjectives in Chinese, such as "large, small, many, few, high, low", which all have the meaning of measurement. Although the number of such ambiguous adjectives is not large, they are frequently used in real text, especially in the texts expressing opinions and emotions. As demonstrated by the experimental results in this paper, the disambiguation of 14 DSAAs can obviously improve the performance of sentiment classification of product reviews.

The task of disambiguating DSAAs is reduced to sentiment classification of nouns. Previous studies classify nouns into three categories: positive, negative and neutral. In contrast, we propose two categories of sentiment expectation

*Most of the work was performed when the author was a student at Peking University.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1191–1199,
Beijing, August 2010

of nouns: positive expectation and negative expectation. This paper presents a novel approach to automatically predict sentiment expectation of nouns. First, we infer the sentiment expectation of a noun by mining the Web with strongly-polar-steering lexico-syntactic patterns. Secondly, we derive the sentiment expectation of a noun from its component characters, which capture the semantic relationship between Chinese words and characters. Finally, a better performance is obtained by combing the two methods. We conduct two types of experiments: the experimental results at the sentence level validate the effectiveness of our approach; the experimental results at the document level confirm the significance of the problem we addressed.

2 Related Work

2.1 Word-level Sentiment Analysis

Recently there has been extensive research in sentiment analysis, for which Pang and Lee (2008) give an in-depth survey of literature. Closer to our study is the large body of work on automatic SO prediction of words (Hatzivassiloglou and McKeown, 1997; Turney and Littman, 2003; Kim and Hovy, 2004; Andreevskaia and Bergler, 2006), but unfortunately they all discard DSAAs in their research. In recent years, some studies go a step further, attaching SO to senses instead of word forms (Esuli and Sebastiani, 2006; Wiebe and Mihalcea, 2006; Su and Markert 2008), but their work is still limited in lexicon out of context.

The most relevant work is Ding et al. (2008), in which DSAAs are named as context dependant opinions. They argue that there is no way to know the SO of DSAAs without prior knowledge, and asking a domain expert to provider such knowledge is scalable. They adopt a holistic lexicon-based approach to solve this problem, which exploits external information and evidences in other sentences and other reviews. On the contrary in this paper, we obtain the prior knowledge of a product by mining the web, and then use such knowledge to determine the SO of DSAAs. The prior knowledge of a product, which is closer to the sentiment expectation of nouns described in this paper, is an important research issue in itself and has many applications in sentiment analysis, as discussed in section 3.2.

2.2 Phrase-level Sentiment Analysis

The disambiguation of DSAAs can also be considered as a problem of phrase-level sentiment analysis. Wilson et al. (2004) present a two-step process to recognize contextual polarity that employs machine learning and a variety of features. Takamura et al. (2006, 2007) propose latent variable model and lexical network to determine SO of phrases, focusing on "noun+adjective" pairs. Their experimental results suggest that the classification of pairs containing ambiguous adjectives is much harder than those with unambiguous adjectives. The above mentioned approaches are all supervised, and need human labeled data for training. In contrast, our method is unsupervised and can overcome the data acquisition bottleneck. Moreover, we focus on the much harder task of disambiguating DSAAs in "noun+adjective" pairs.

2.3 Pattern-based Method

Previous studies have applied pattern-based method to sentiment analysis (Riloff and Wiebe, 2003; Wiebe et al., 2004; Riloff et al., 2005; Wiebe and Mihalcea, 2006; Andreevskaia and Berger; 2006). The differences with our method lie in two aspects: the used resources (corpus versus web) and the research target (subjective expressions versus sentiment expectation).

2.4 Character-based Method

Chinese characters carry semantic information that is indicative of semantic properties of words. Previous studies have exploited the character-based model to predict the semantic categories of Chinese unknown words (Chen, 2004; Lu, 2007). Yuen et al. (2004) presents a method to infer the SO of a Chinese word from its statistical association with strong-polarized characters rather than with strong-polarized words. The work by Ku et al. (2006) is similar to ours because they also define the sentiment score of a word by its composite characters. However, their algorithm is based only on frequency, while we exploit point mutual information that can capture the character-sentiment association.

3 Determining SO of Adjective by Target Noun

3.1 Classification of DSAAs

The frequently used DSAAs are given below. We group them into two categories: positive-like adjectives and negative-like adjectives. These adjectives are neutral out of context, but positive or negative emotion will be evoked when they co-occur with some target nouns, making it hard to assign each to a fixed sentiment category in lexicon.

(1) Positive-like adjectives (*Pa*) = {大 da|large, 多 duo|many, 高 gao|high, 厚 hou|thick, 深 shen|deep, 重 zhong|heavy, 巨大 ju-da|huge, 重大 zhong-da|great}

(2) Negative-like adjectives (*Na*) ={小 xiao|small, 少 shao |few, 低 di|low, 薄 bao|thin, 浅 qian|shallow, 轻 qing|light}

3.2 Sentiment Expectation of Noun

The SO of most DSAAs can be determined by target nouns in noun-adjective phrases, as shown in Table 1. For example, the word "high" has a positive orientation when the target noun is "salary" but a negative orientation when the target noun is "price". Therefore, the task can be reduced to sentiment classification of nouns.

Positive 潜力大|potential is great 工资高|salary is high	Negative 潜力小|potential is small 工资低|salary is low
Negative 压力大|pressure is big 价格高|price is high	Positive 压力小|pressure is small 价格低|price is low

Table 1: The SO of DSAAs in noun-adjective phrases

In previous research, the SO of nouns is classified into three categories: positive, negative and neutral. Accordingly, "压力 ya-li|pressure" will be assigned as negative and "潜力 qian-li|potential" as positive, while "工资 gong-zi|salary" and "价格 jia-ge|price" will be assigned as neutral, as the two terms are objective and cannot evoke positive or negative emotion. Different from the traditional classification scheme, we propose sentiment expectation and classify nouns into two categories: positive expectation and negative expectation. For a positive expectation noun, people usually expect the thing referred to by the noun to be bigger, higher or happen frequently. On the contrary, for a negative expectation noun, people usually expect the thing referred to by the noun to be smaller, lower or don't happen . For example, "价格 jia-ge|price" is a negative expectation noun, as most people in most cases expect that the product prices become low, whereas "工资 gong-zi|salary" is a positive expectation noun, as most people in most cases expect that their salaries become high. The relationship between traditional SO and sentiment expectation can be defined as: positive (negative) terms correspond to positive (negative) expectation terms, but some neutral terms may also carry positive (negative) expectation.

Su and Markert (2008) argue that polarity can also be attached to objective words. The difference with our scheme is that, for example, "价格 jia-ge|price" is attached to negative expectation in our scheme while is still neutral in Su and Markert's method.

The distinction between positive and negative expectation nouns is vital to determine the SO of some phrases. Using it to disambiguate DSAAs is a good example. Another application is the phrase containing verbs with the meaning of status change. For example, "工资上涨了|salary has been raised" will evoke positive emotion, while "价格上涨了 jiage-shangzhang-le|prices have gone up" will evoke negative emotion. As far as we are aware, this is the first sentiment analysis scheme that tries to exploit people's expectation towards nouns.

3.3 Determination of DSAAs

The SO of DSAAs in a given phrase can be calculated by Eq. (1).

$$C(a) = \begin{cases} 1 & \text{if a is positive-like} \\ -1 & \text{if a is negative-like} \end{cases} \quad (1)$$

$$C(n) = \begin{cases} 1 & \text{if n is positive expectation} \\ -1 & \text{if n is negative expectation} \end{cases}$$

$$SO(a)=C(a)*C(n)$$

If adverb="不 bu|not", $SO(a)= -SO(a)$

Where C(a) denotes the category of DSAAs; C(n) denotes the sentiment expectation of nouns; SO(a) is the SO of DSAAs in a give noun-adjective phrase. When the adverb is the negation term "不 bu|not", the SO is reversed.

4 Predicting Sentiment Expectation of Noun

4.1 Pattern-based Prediction Using a Web Search Engine

In natural language, there are some lexico-syntactic patterns that people frequently use when they express their opinion about something. For example:

(3) 工资*有点*低 | Salary is *a little* low.
(4) 价格*有点*高| Price is *a little* high.

The pattern "<n> 有点 <a>" carries a strong negative association in Chinese language. When a man is saying "工资有点低| Salary is a little low", it indicates that he wishes his "工资 |salary" to be raised. On the contrary, when a man is saying "价格有点高| price is a little high", it indicates that he wishes "价格 |price" to go down. As a result, "工资 |salary" has positive expectation while "价格 |price" has negative expectation.

With the rapid development and expansion of the internet, Web has become an important medium for people to post their ideas. The opinions expressed on the Web reflect the common cognition shared by collection of people in a culture. Therefore, using a Web search engine with the strong-polar-steering lexico-syntactic patterns as queries, we can infer the sentiment expectation of a noun, by calculating its statistical association with positive and negative hits.

As an example, using the search engine Baidu[2] with the pattern "<n> 有点 <a>" as queries, we obtain the following hits:

(5) 工资*有点*低 | Salary is a little low. (2890 hits)
工资*有点*高 | Salary is a little high (67 hits)
(6) 价格*有点*高 | Price is a little high. (19400 hits)
价格*有点*低 | Price is a little low. (1080 hits)

The more than 40 times more numerous hits for "工资有点低 |Salary is a little low" indicate that that "工资|salary" is a positive expectation noun. For the same reason, we can infer that "价格 |price" has negative expectation.

DSAAs are classified into two opposite sets *Pa* and *Na*, as listed in (1) and (2) respectively. Here two-character adjectives ("巨大 |huge" and "重大 |great") are discarded. Four types of lexico-syntactic patterns, which are also classified into two opposite sets in consistent with *Pa* and *Na*, are used in this paper, as listed in Table 2. These patterns were manually designed, inspired by linguistic knowledge and after a deep investigation on the Web.

Pos. expectation patterns	Neg. expectation patterns
1) <n> 有点 *Na* n is a little *Na*	1) <n> 有点 *Pa* n is a little *Pa*
2) <n> 有点儿 *Na* n is a little *Na*	2) <n> 有点儿 *Pa* n is a little *Pa*
3) <n> *Na*, 怎么办 n is *Na*, what should we do?	3) <n> *Pa*, 怎么办 n is *Pa*, what should we do?
4) 嫌 <n> *Na* n is too *Na*	4) 嫌 <n> *Pa* n is too *Pa*

Table 2: The lexico-syntactic patterns

Here the noun (n) in these patterns was instantiated by 9,468 nouns in our collected data. A noun has together 48 patterns, 24 positive and 24 negative ones. For each noun, we obtain the hits of both positive and negative expectation patterns, using the search engine Baidu. The sentiment expectation of a noun is acquired by Eq. (2) and Eq. (3), where the magnitude of $PT\_SO(n)$ can be considered as the strength of sentiment expectation.

$$PT\_SO(n) = \sum_{b \in Na} \sum_{i=1}^{4} PositivePatternHit_i(n, b) - \sum_{a \in Pa} \sum_{i=1}^{4} NegativePatternHit_i(n, a) \quad (2)$$

$$\text{n is } \begin{cases} \text{positive expectation} & \text{if } PT\_SO(n)>0 \\ \text{negative expectation} & \text{if } PT\_SO(n)<0 \\ \text{not predicted} & \text{if } PT\_SO(n)=0 \end{cases} \quad (3)$$

Table 3 gives some nouns with sentiment expectation predicted by the pattern-based method, descending (the left column) and ascending (the right column) by the absolute value of $PT\_SO(n)$. Most words (9 out of 10) are correctly predicted, demonstrating that the result of pattern-based method is promising. The only wrong predicted noun is "感觉 |feeling", due to the fact that most instances of it on the Web data are used as verb rather than noun, like "感觉有点大| I think it is large".

[2] http://baidu.com.cn.

Positive expectation	Negative expectation
Noun ($PT\_SO(n)$)	Noun ($PT\_SO(n)$)
钱|money (31349)	温度|temperature(-111576)
工资|wage (26311)	噪音|noise (-45790)
感觉|feeling (20102)	价格|price (-25653)
收入|income(19429)	代价|cost (-22051)
官|officer (10630)	血压|blood pressure (-21788)

Table 3: Examples of nouns with sentiment expectation predicted by the pattern-based method

4.2 Character-based Derivation Using Sentiment Lexicons

But the sentiment expectation of some nouns cannot be predicted with the pattern-based method, mainly due to the reason that these nouns don't occur in the listed patterns in Table 2. An alternate way is to exploit the semantic knowledge of Chinese characters. It is assumed that there is a strong association between the sentiment category of a word and its component characters. For example, the three words "罪恶 zui'e|evil, 罪行 zuixing|crime, 罪过 zuiguo|fault", which all contain the character "罪 zui|sin" that carries negative meaning, are all negative expectation nouns.

First, we compute the character-word sentiment association by the following PMI formula, based on a sentiment lexicon:

$$PMI(c, Positive) = \log \frac{P(c, Positive)}{P(c)P(Positive)}$$
$$PMI(c, Negative) = \log \frac{P(c, Negative)}{P(c)P(Negative)} \quad (4)$$
$$SO(c) = PMI(c, Positive) - PMI(c, Negative)$$

Where $P(c, Positive)$ is the probability of a character c in the positive category; $P(c)$ is the probability of a character c in the sentiment lexicon; $P(Positive)$ is the probability of the positive category in the sentiment lexicon. $PMI(c, Negative)$ has the similar meaning. Probabilities are estimated according to the maximum likelihood principle.

The open language resources for Chinese sentiment analysis are quite limited. We selected the following two sentiment lexicons.

Sentiment HowNet. HowNet has published the Chinese vocabulary for sentiment analysis[3], which was manually constructed. The positive category contains 4,566 words and the negative category contains 4,370 words.

Sentiment BaiduHit. In our collected data, we extracted 9,468 nouns. Using the pattern-based method we acquired sentiment expectation of these nouns, where 2,530 ones were assigned as positive expectation, 1,837 ones as negative expectation and 5,101 ones were not predicted. It is assumed that most nouns are correctly predicted. These nouns with their sentiment expectation constitute the lexicon of Sentiment BaiduHit, which is automatically constructed.

Combining HowNet and BaiduHit. Most sentiment characters derived from HowNet have adjective property, since most words in Sentiment HowNet are adjectives. On the contrary, most sentiment characters derived from BaiduHit have noun property. Therefore, the combination of the two lexicons can cover more characters. As Sentiment HowNet is manually compiled, the sentiment characters derived from it should be more reasonable than those from BaiduHit. When combining the two lexicons in computing character polarity, we assign a high priority to HowNet. Only when a character is out of vocabulary in HowNet, we resort to BaiduHit.

Then, we acquire the sentiment category of a word by computing the following equation. Let a word consist of n characters $w = c_1, c_2, \ldots c_n$, the sentiment category of the word is calculated by the average sentiment value of its component characters:

$$CH\_SO(w) = \frac{1}{n}\sum_{i=1}^{n} SO(c_i) \quad (5)$$

$$\text{w is } \begin{cases} \text{positive expectation} & \text{if } CH\_SO(w)>0 \\ \text{negative expectation} & \text{if } CH\_SO(w)<0 \\ \text{neutral} & \text{if } CH\_SO(w)=0 \end{cases} \quad (6)$$

We acquired *sentiment expectation* of 9,468 nouns in our collected data, based on Sentiment HowNet, Sentiment BaiduHit, and the combination of the two lexicons, respectively.

Table 6 gives examples of nouns with sentiment expectation acquired by the character-based method combining the two lexicons of HowNet and BaiduHit, descending (the left column) and ascending (the right column) by the absolute value of $CH\_SO(w)$.

[3] http://www.keenage.com/html/c_index.html.

Positive expectation	Negative expectation
Noun($CH\_SO(w)$)	Noun($CH\_SO(w)$)
美称|good name (3.23)	灰|ash (-3.22)
健美|health (3.06)	毛|gross (-2.93)
香|fragrance (3.05)	税|tax (-2.89)
美方|U.S.A (2.98)	毛病|fault (-2.84)
职称|title (2.64)	毒|poison (-2.82)

Table 4: Example of nouns with sentiment expectation predicted by the character-based method

4.3 Integrating Pattern-based Prediction and Character-based Derivation

The two methods of pattern-based prediction and character-based derivation have complementary properties. The pattern-based method concentrates on a word's usage on the Web, whereas the character-based method focuses on the internal structure of a word. So the two methods can be integrated to get better performance. The results using pattern-based method are much better than character-based method, as illustrated in Table 3 and Table 4. So in the integrated scheme, we give a high priority to pattern-based method. The pattern-based approach is mainly used, and only when the value of $|PT\_SO(n)|$ is smaller than a threshold r, the character-based method is adopted. Because when the value of $|PT\_SO(n)|$ is very small, it could be caused by random noises on the Web. We set r to 9 according to empirical analysis in the development data.

5 Experiments

5.1 Sentiment Analysis at Sentence Level

5.1.1 Data

We collected data from two sources. The main part was extracted from Xinhua News Agency of Chinese Gigaword (Second Edition) released by LDC. The texts were automatically word-segmented and POS-tagged using the open software ICTCLAS[4]. In order to concentrate on the disambiguation of DSAAs, and reduce the noise introduced by the parser, we extracted sentences containing strings in pattern of (7), where the target noun is modified by the adjective in most cases.

(7) noun+adverb+adjective (adjective∈DSAAs)
e.g. 成本/n 较/d 低/a | the cost is low.

Another small part of data was extracted from the Web. Using the search engine Google[5], we searched the queries as in (8):

(8) 很| very+ adjective (adjective∈DSAAs)

From the returned snippets, we manually picked out some sentences that contain the strings of (7). Also, the sentences were automatically word-segmented and POS-tagged using ICTCLAS.

DSAAs in the data were assigned as positive, negative or neutral, independently by two annotators. Since we focus on the distinction between positive and negative categories, the neutral instances were removed. Table 5 gives statistics of the data, and the inter-annotator agreement is in a high level with a kappa of 0.91. After cases with disagreement were negotiated between the two annotators, a gold standard annotation was agreed upon. In this paper, 3066 instances were divided randomly into three parts, 1/3 of which were used as the development data, and 2/3 were the test data.

Most of the data has been used as the benchmark dataset of SemEval-2010 task 18 "disambiguating sentiment ambiguous adjectives" (Wu and Jin, 2010), and so it can be downloaded freely for research.

	Pos#	Neg#	Total#
Pos#	1280	58	1338
Neg#	72	1666	1738
Total#	1352	1724	3066

Table 5: The statistics of DSAAs data

5.1.2 Baseline

We conducted two types of baseline.

Simple Baseline. Not considering the context, assign all positive-like adjectives as positive, and all negative-like adjectives as negative.

HowNet Baseline. Acquiring SO of nouns from Sentiment HowNet, the polarity of DSAAs is computed by Eq. (1).

5.1.3 Methods

Pattern-based method. Acquiring sentiment expectation of nouns using the pattern-based method, the polarity of DSAAs is computed by Eq.(1).

[4] http://www.ictclas.org/.

[5] http://www.google.com/.

Character-based method. Acquiring sentiment expectation of nouns using the character-based method, based on Sentiment HowNet, Sentiment BaiduHit and the combination of the two lexicons respectively, the polarity of DSAAs is computed by Eq.(1).

Integrated method. Acquiring sentiment expectation of nouns by integrating pattern-based and character-based methods, the polarity of DSAAs is computed by Eq. (1).

5.1.4 Results

Table 6 gives the experimental results at sentence level with different methods.

Methods	Pre.	Rec.	F
Simple Baseline	61.20	61.20	61.20
HowNet Baseline	97.58	9.88	17.94
Pattern-based	75.83	71.67	73.69
Character-based (HowNet)	69.89	69.37	69.63
Character-based (BaiduHit)	68.66	68.59	68.62
Character-based (Combined)	71.01	70.94	70.97
Integrated method	**78.52**	**78.52**	**78.52**

Table 6: The experimental results at sentence level

As for the simple baseline, both the precision and recall are low, suggesting that DSAAs cannot be neglected for sentiment analysis in a real-world application.

The HowNet baseline achieves a quite high precision of 97.58%, but a rather poor recall of 9.88%, suggesting that SO of nouns described in traditional sentiment lexicon, like HowNet, cannot effectively disambiguate DSAAs.

The proposed methods in this paper all yield results that are substantially better than two types of baseline. The pattern-based method, as straightforward as it is, achieves promising result with an f-score of 73.69%, which is 12.49% higher than the simple baseline. The pattern-based method outperforms the character-based method (combined) by 4.82% in precision and 0.73% in recall. The performance of the character-based method based on Sentiment BaiduHit is competitive with that based on Sentiment HowNet, which again proves the effectiveness of the pattern-based method. The character-based method combining the two lexicons outperforms each lexicon with small improvement. The approach integrating pattern-based and character-based methods outperforms each method in isolation, achieving an f-score of 78.52% that is 17.32% higher than the simple baseline and 60.58% higher than HowNet baseline.

5.2 Sentiment Analysis at Document Level

5.2.1 Data

We also investigated the impact of disambiguating DSAAs on the sentiment classification of product reviews. Following the work of Wan (2008), we selected the same dataset. The dataset contains 886 Chinese product reviews, which are manually annotated with polarity labels: positive or negative. Also, the files are automatically word-segmented and POS-tagged using ICTCLAS. We extracted the files that contain the following strings, where the nouns are modified by DSAAs in most cases.

(9) noun+adjective (adjective∈DSAAs)
noun+adverb+adjective
noun+adverb+adverb+adjective.

We obtained 212 files, up to 24% of the overall data, suggesting again that DSAAs are frequently used in product reviews and cannot be avoided in a real-world application.

5.2.2 Methods

Our goal is not to propose a new method, but instead to test the performance gain by adding the disambiguation of DSAAs. We adopted the same algorithm with Wan (2008), and also used Sentiment-HowNet. But in our experiment, *Negation_Dic* contains only one term "不 bu|not", for the sake of repeatable experiments.

The baseline algorithm is illustrated by the non-italic part in Figure 1, where we set the same parameters with Wan's approach: *PosValue*=1, *NegValue*=-2, q=2, ρ=2.

We added the disambiguation of DSAAs to the algorithm, as illustrated by the italic part in Figure 1. When a word is a DSAA, compute its SO with the proposed integrated method, rather than using its prior polarity specified in HowNet. For *Dy_PosValue* and *Dy_NegValue*, we first set *Dy_PosValue*=1 and *Dy_NegValue*=-2, just the same as *PosValue* and *NegValue*. In the second attempt, in order to further intensify the polarity of DSAAs, we set *Dy_PosValue*=1.5 and *Dy_NegValue*=-2.5. Other parameters were set the same as baseline.

Algorithm Compute_SO:

1. Tokenize document d into sentence set S, and each sentence s∈S is tokenized into word set Ws;
2. For any word w in a sentence s∈S, compute its value SO(w) as follows:
   1) if w∈DSAAs, compute SO(w) with the integrated method.
      If SO(w)=1, SO(w)=Dy_PosValue;
      If SO(w)=-1, SO(w)=Dy_NegValue;
   2) if w∈Positive_Dict, SO(w)=PosValue;
   3) If w∈Negative_Dict, SO(w)=NegValue;
   4) Otherwise, SO(w)=0;
   5) Within the window of q words previous to w, if there is a term w'∈Negation_Dict, SO(w)= –SO(w);
   6) Within the window of q words previous to w, if there is a term w'∈Intensifier_Dict, SO(w) =ρ×SO(w);
3. $S(d) = \sum_{s \in S} \sum_{w \in Ws} SO(w)$

Figure 1: Algorithm of computing SO of documents

5.2.3 Results

Adding the disambiguation of *DSAAs*, the performance of sentiment classification of 212 product reviews was significantly improved, as shown in Table 7.

		Baseline	DSAAs (1, -2)	DSAAs (1.5, -2.5)
Pos.	Pre.	75.89	77.50	76.61
	Rec.	78.70	86.11	87.96
	F	77.27	81.58	81.90
Neg.	Pre.	87.01	88.46	87.06
	Rec.	64.42	66.35	71.15
	F	74.03	75.82	78.31
Total	MacroF	75.62	78.60	**80.06**
	Accu.	71.70	76.42	**79.72**

Table 7: The experimental results at document level

As an example, the following review, which consists of only one sentence, is correctly classified as positive by DSAAs method, but is classified as negative by the baseline approach.

(10) 体积小，重量轻，携带很方便。
| Small size, light weight, and easy to carry.

According to HowNet, as shown in Table 8, the sentence contains two negative words "小|small" and "轻|light" and one positive word "方便 fangbian|easy", resulting the overall negative prediction. In our approach, "体积 tiji|size" and "重量 zhongliang|weight" are assigned as negative expectation, and consequently both "体积小|small size" and "重量轻|light weight" have positive meaning, resulting the overall positive prediction.

Pos.	大 |large, 高 |high, 厚 |thick, 深 |deep, 重|heavy, 重大|great
Neg.	小 |small, 低 |low, 薄 |thin, 浅 |shallow, 轻|light
OOV	多|many, 少|few, 巨大|huge

Table 8: The SO of DSAAs described in HowNet

Adding the disambiguation of DSAAs, our method obviously outperforms the baseline by 4.44% in f-score and 8.02% in accuracy. The improvement in recall is especially obvious. When intensifying the polarity of DSAAs by setting *Dy_PosValue*=1.5 and *Dy_NegValue*=-2.5, the recall is improved by 9.26% for positive category and 6.73% for negative category.

6 Conclusion and Future Work

This paper presents a knowledge-based unsupervised method to automatically disambiguate dynamic sentiment ambiguous words, focusing on 14 DSAAs. We exploit pattern-based and character-based methods to infer sentiment expectation of nouns, and then determine the polarity of DSAAs based on the nouns. For the sentiment analysis at sentence level, our method achieves promising result that is significantly better than two types of baseline, which validates the effectiveness of our approach. We also apply the disambiguation of 14 DSAAs to the sentiment classification of product reviews, resulting obvious improvement in performance, which proves the significance of the issue.

There leaves room for improvement. Our future work will explore more contextual information in disambiguating DSAAs. In addition, we will find out new methods to reduce noises when mining the Web to infer sentiment expectation of nouns. Discovering the lexico-syntactic patterns for sentiment expectation of nouns automatically or semi-automatically with bootstrapping method is also a challenging direction.

Acknowledgments

This work was supported by National Natural Science Foundation of China (No. 60703063) and National Social Science Foundation of China (No. 08CYY016).

References

Andreevskaia A. and Bergler S. 2006. Sentiment tagging of adjectives at the meaning level. *The 19th Canadian Conference on Artificial Intelligence.*

Andreevskaia, A. and Bergler, S. 2006. Mining WordNet for fuzzy sentiment: Sentiment tag extraction from WordNet glosses. *Proceedings of EACL 2006.*

Chen, C-J. 2004. Character-sense association and compounding template similarity: automatic semantic classification of Chinese compounds. *Proceedings of the 3rd workshop on Chinese language processing.*

Ding X., Liu B. and Yu, P. 2008. A holistic lexicon-based approach to opinion mining. *Proceedings of WSDM'08.*

Esuli, A. and Sebastiani, F. 2006. SentiWordNet: a publicly available lexical resource for opinion mining. *Proceedings of LREC'06.*

Hatzivassiloglou, V. and McKeown, K. 1997 Predicting the semantic orientation of adjectives. *Proceedings of ACL'97.*

Kim, S and Hovy, E. 2004. Determining the sentiment of opinions. *Proceedings of COLING'04.*

Ku, L, Liang Y. and Chen, H. 2006. Opinion extraction, summarization and tracking in news and blog corpora. *Proceedings of AAAI-2006 Spring Symposium on Computational Approaches to Analyzing Weblogs.*

Lu X-F, 2007. Hybrid models for semantic classification of Chinese unknown words. *Proceedings of NAACL HLT'07.*.

Pang, B. and Lee, L. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval.*

Riloff, E. and Wiebe, J. 2003. Learning Extraction Patterns for Subjective Expressions. *Proceedings of EMNLP'03.*

Riloff, E., Wiebe, J. and Phillips, W. 2005. Exploiting Subjectivity Classification to Improve Information Extraction. *Proceedings of AAAI'05.*

Su, F. and Markert, K. 2008. From words to senses: a case study of subjectivity recognition. *Proceedings of COLING'08.*

Takamura, H., Inui,T. and Okumura, M. 2006. Latent Variable Models for Semantic Orientations of phrases. *Proceedings of EACL'06.*

Takamura, H., Inui,T. and Okumura, M. 2007. Extracting Semantic Orientations of Phrases from Dictionary. *Proceedings of NAACL HLT '07.*

Turney, P. and Littman, M. 2003. Measuring praise and criticism: inference of semantic orientation from association. *ACM transaction on information systems.*

Wan, X. 2008. Using Bilingual Knowledge and Ensemble Techniques for Unsupervised Chinese Sentiment Analysis. *Proceedings of EMNLP'08.*

Wiebe, J. and Mihalcea, R. 2006. Word sense and subjectivity. *Proceedings of ACL'06.*

Wiebe, J., Wilson, T., Bruce, R., Bell, M. and Martin, M. 2004. Learning Subjective Language. *Computational Linguistics.*

Wilson, T., Wiebe, J. and Hoffmann, P. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. *Proceedings of HLT/EMNLP'05.*

Wu, Y. and Jin, P. 2010. SemEval-2010 task 18: disambiguating sentiment ambiguous adjectives. *Proceedings of SemEval 2010.*

Yuen R., Chan T., Lai T., Kwong O., T'sou B. 2004. Morpheme-based derivation of bipolar semantic orientation of Chinese words. *Proceedings of COLING'04.*

Joint Tokenization and Translation

Xinyan Xiao † **Yang Liu** † **Young-Sook Hwang** ‡ **Qun Liu** † **Shouxun Lin** †

†Key Lab. of Intelligent Info. Processing
Institute of Computing Technology
Chinese Academy of Sciences
{xiaoxinyan,yliu,liuqun,sxlin}@ict.ac.cn

‡HILab Convergence Technology Center
C&I Business
SKTelecom
yshwang@sktelecom.com

Abstract

As tokenization is usually ambiguous for many natural languages such as Chinese and Korean, tokenization errors might potentially introduce translation mistakes for translation systems that rely on 1-best tokenizations. While using lattices to offer more alternatives to translation systems have elegantly alleviated this problem, we take a further step to tokenize and translate jointly. Taking a sequence of atomic units that can be combined to form words in different ways as input, our joint decoder produces a tokenization on the source side and a translation on the target side simultaneously. By integrating tokenization and translation features in a discriminative framework, our joint decoder outperforms the baseline translation systems using 1-best tokenizations and lattices significantly on both Chinese-English and Korean-Chinese tasks. Interestingly, as a tokenizer, our joint decoder achieves significant improvements over monolingual Chinese tokenizers.

1 Introduction

Tokenization plays an important role in statistical machine translation (SMT) because tokenizing a source-language sentence is always the first step in SMT systems. Based on the type of input, Mi and Huang (2008) distinguish between two categories of SMT systems : *string-based* systems (Koehn et al., 2003; Chiang, 2007; Galley et al., 2006; Shen et al., 2008) that take a string as input and *tree-based* systems (Liu et al., 2006; Mi et al., 2008) that take a tree as input. Note that a tree-based system still needs to first tokenize the input sentence and then obtain a parse tree or forest of the sentence. As shown in Figure 1(a), we refer to this pipeline as **separate** tokenization and translation because they are divided into single steps.

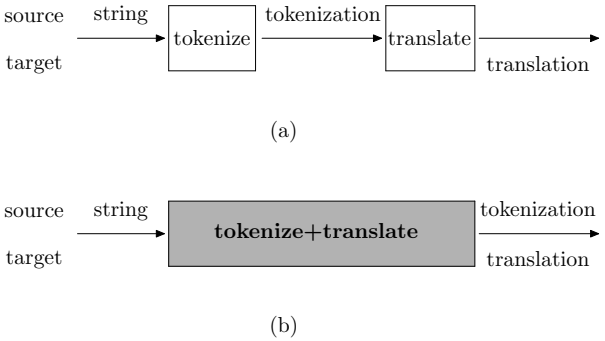


Figure 1: (a) Separate tokenization and translation and (b) joint tokenization and translation.

As tokenization for many languages is usually ambiguous, SMT systems that separate tokenization and translation suffer from a major drawback: tokenization errors potentially introduce translation mistakes. As some languages such as Chinese have no spaces in their writing systems, how to segment sentences into appropriate words has a direct impact on translation performance (Xu et al., 2005; Chang et al., 2008; Zhang et al., 2008). In addition, although agglutinative languages such as Korean incorporate spaces between "words", which consist of multiple morphemes, the granularity is too coarse and makes the training data

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1200–1208,
Beijing, August 2010

considerably sparse. Studies reveal that segmenting "words" into morphemes effectively improves translating morphologically rich languages (Oflazer, 2008). More importantly, a tokenization close to a gold standard does not necessarily leads to better translation quality (Chang et al., 2008; Zhang et al., 2008). Therefore, it is necessary to offer more tokenizations to SMT systems to alleviate the tokenization error propagation problem. Recently, many researchers have shown that replacing 1-best tokenizations with lattices improves translation performance significantly (Xu et al., 2005; Dyer et al., 2008; Dyer, 2009).

We take a next step towards the direction of offering more tokenizations to SMT systems by proposing **joint** tokenization and translation. As shown in Figure 1(b), our approach tokenizes and translates jointly to find a tokenization and a translation for a source-language string simultaneously. We integrate translation and tokenization models into a discriminative framework (Och and Ney, 2002), within which tokenization and translation models interact with each other. Experiments show that joint tokenization and translation outperforms its separate counterparts (1-best tokenizations and lattices) significantly on the NIST 2004 and 2005 Chinese-English test sets. Our joint decoder also reports positive results on Korean-Chinese translation. As a tokenizer, our joint decoder achieves significantly better tokenization accuracy than three monolingual Chinese tokenizers.

2 Separate Tokenization and Translation

Tokenization is to split a string of characters into meaningful elements, which are often referred to as words. Typically, machine translation separates tokenization from decoding as a preprocessing step. An input string is first preprocessed by a tokenizer, and then is translated based on the tokenized result. Take the SCFG-based model (Chiang, 2007) as an example. Given the character sequence of Figure 2(a), a tokenizer first splits it into the word sequence as shown in Figure 2(b), then the decoder translates the word sequence using the rules in Table 1.

This approach makes the translation process simple and efficient. However, it may not be

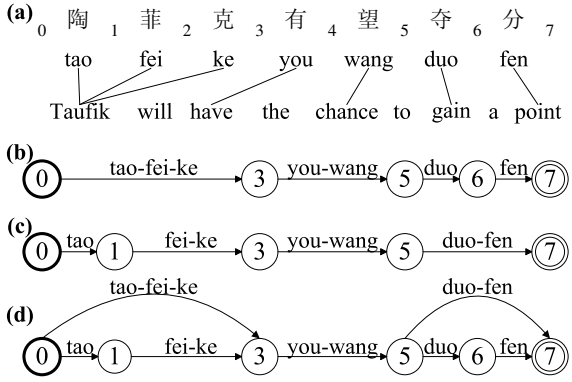


Figure 2: Chinese tokenization: (a) character sequence; (b) and (c) tokenization instances; (d) lattice created from (b) and (c). We insert "-" between characters in a word just for clarity.

r_1	*tao-fei-ke* $\rightarrow$*Taufik*
r_2	*duo fen* $\rightarrow$ *gain a point*
r_3	x_1 *you-wang* $x_2 \rightarrow x_1$ *will have the chance to* x_2

Table 1: An SCFG derivation given the tokenization of Figure 2(b).

optimal for machine translation. Firstly, optimal granularity is unclear for machine translation. We might face severe data sparseness problem by using large granularity, while losing much useful information with small one. Consider the example in Figure 2. It is reasonable to split *duo fen* into two words as *duo* and *fen*, since they have one-to-one alignments to the target side. Nevertheless, while *you* and *wang* also have one-to-one alignments, it is risky to segment them into two words. Because the decoder is prone to translate *wang* as a verb *look* without the context *you*. Secondly, there may be tokenization errors. In Figure2(c), *tao fei ke* is recognized as a Chinese person name with the second name *tao* and the first name *fei-ke*, but the whole string *tao fei ke* should be a name of the Indonesian badminton player.

Therefore, it is necessary to offer more tokenizations to SMT systems to alleviate the tokenization error propagation problem. Recently, many researchers have shown that replacing 1-best tokenizations with lattices improves translation performance significantly. In this approach, a lattice compactly encodes many tokenizations and is fixed before decoding.

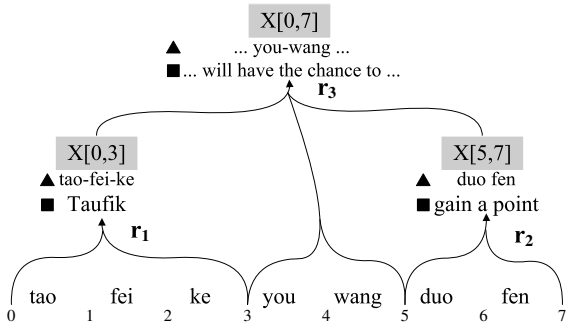


Figure 3: A derivation of the joint model for the tokenization in Figure 2(b) and the translation in Figure 2 by using the rules in Table 1. ▲ means tokenization while ■ represents translation.

3 Joint Tokenization and Translation

3.1 Model

We take a next step towards the direction of offering more tokenizations to SMT systems by proposing **joint** tokenization and translation. As shown in Figure 1(b), the decoder takes an untokenized string as input, and then tokenizes the source side string while building the corresponding translation of the target side. Since the traditional rules like those in Table 1 natively include tokenization information, we can directly apply them for simultaneous construction of tokenization and translation by the source side and target side of rules respectively. In Figure 3, our joint model takes the character sequence in Figure 2(a) as input, and synchronously conducts both translation and tokenization using the rules in Table 1.

As our model conducts tokenization during decoding, we can integrate tokenization models as features together with translation features under the discriminative framework. We expect tokenization and translation could collaborate with each other. Tokenization offers translation with good tokenized results, while translation helps tokenization to eliminate ambiguity. Formally, the probability of a derivation D is represented as

$$P(D) \propto \prod_i \phi_i(D)^{\lambda_i} \quad (1)$$

where ϕ_i are features defined on derivations including translation and tokenization, and λ_i are feature weights. We totally use 16 features:

- 8 traditional translation features (Chiang, 2007): 4 rule scores (direct and reverse translation scores; direct and reverse lexical translation scores); language model of the target side; 3 penalties for word count, extracted rule and glue rule.
- 8 tokenization features: maximum entropy model, language model and word count of the source side (Section 3.2). To handle the Out Of Vocabulary (OOV) problem (Section 3.3), we also introduce 5 OOV features: OOV character count and 4 OOV discount features.

Since our model is still a string-based model, the CKY algorithm and cube pruning are still applicable for our model to find the derivation with max score.

3.2 Adding Tokenization Features

Maximum Entropy model (ME). We first introduce ME model feature for tokenization by casting it as a labeling problem (Xue and Shen, 2003; Ng and Low, 2004). We label a character with the following 4 types:

- **b**: the **b**egin of a word
- **m**: the **m**iddle of a word
- **e**: the **e**nd of a word
- **s**: a **s**ingle-character word

Taking the tokenization *you-wang* of the string *you wang* for example, we first create a label sequence *b e* for the tokenization *you-wang* and then calculate the probability of tokenization by

$$\begin{aligned} &P(\textit{you-wang} \mid \textit{you wang}) \\ &= P(\mathrm{b\ e} \mid \textit{you wang}) \\ &= P(\mathrm{b} \mid \textit{you}, \textit{you wang}) \\ &\quad \times P(\mathrm{e} \mid \textit{wang}, \textit{you wang}) \end{aligned}$$

Given a tokenization w_1^L with L words for a character sequence c_1^n, we firstly create labels l_1^n for every characters and then calculate the probability by

$$P(w_1^L|c_1^n) = P(l_1^n|c_1^n) = \prod_{i=1}^{n} P(l_i|c_i, c_1^n) \quad (2)$$

Under the ME framework, the probability of assigning the character c with the label l is represented as:

$$P(l|c, c_1^n) = \frac{\exp[\sum_i \lambda_i h_i(l, c, c_1^n)]}{\sum_{l'} \exp[\sum_i \lambda_i h_i(l', c, c_1^n)]} \quad (3)$$

where h_i is feature function, λ_i is the feature weight of h_i. We use the feature templates the same as Jiang et al., (2008) to extract features for ME model. Since we directly construct tokenization when decoding, it is straight to calculate the ME model score of a tokenization according to formula (2) and (3).

Language Model (LM). We also use the n-gram language model to calculate the probability of a tokenization w_1^L:

$$P(w_1^L) = \prod_{i=1}^{L} P(w_i|w_{i-n+1}^{i-1}) \quad (4)$$

For instance, we compute the probability of the tokenization shown in Figure 2(b) under a 3-gram model by

$$\begin{aligned} & P(\textit{tao-fei-ke}) \\ \times & P(\textit{you-wang} \mid \textit{tao-fei-ke}) \\ \times & P(\textit{duo} \mid \textit{tao-fei-ke}, \textit{you-wang}) \\ \times & P(\textit{fen} \mid \textit{you-wang}, \textit{duo}) \end{aligned}$$

Word Count (WC). This feature counts the number of words in a tokenization. Language model is prone to assign higher probabilities to short sentences in a biased way. This feature can compensate this bias by encouraging long sentences. Furthermore, using this feature, we can optimize the granularity of tokenization for translation. If larger granularity is preferable for translation, then we can use this feature to punish the tokenization containing more words.

3.3 Considering All Tokenizations

Obviously, we can construct the potential tokenizations and translations by only using the extracted rules, in line with traditional translation decoding. However, it may limits the potential tokenization space. Consider a string *you wang*. If *you-wang* is not reachable by the extracted rules, the tokenization *you-wang* will never be considered under this way. However, the decoder may still create a derivation by splitting the string as small as possible with tokenization *you wang* and translating *you* with *a* and *wang* with *look*, which may hurt the translation performance. This case happens frequently for named entity especially. Overall, it is necessary to assure that the decoder can derive all potential tokenizations (Section 4.1.3).

To assure that, when a span is not tokenized into a single word by the extracted rules, we will add an operation, which is considering the entire span as an OOV. That is, we tokenize the entire span into a single word with a translation that is the copy of source side. We can define the set of all potential tokenizations $\tau(c_1^n)$ for the character sequence c_1^n in a recursive way by

$$\tau(c_1^n) = \bigcup_{i}^{n-1} \{\tau(c_1^i) \bigotimes \{w(c_{i+1}^n)\}\} \quad (5)$$

here $w(c_{i+1}^n)$ means a word contains characters c_{i+1}^n and $\bigotimes$ means the times of two sets. According to this recursive definition, it is easy to prove that all tokenizations is reachable by using the glue rule ($S \Rightarrow SX, SX$) and the added operation. Here, glue rule is used to concatenate the translation and tokenization of the two variables S and X, which acts the role of the operator $\bigotimes$ in equation (5).

Consequently, this introduces a large number of OOVs. In order to control the generation of OOVs, we introduce the following OOV features:

OOV Character Count (OCC). This feature counts the number of characters covered by OOV. We can control the number of OOV characters by this feature. It counts 3 when *tao-fei-ke* is an OOV, since *tao-fei-ke* has 3 characters.

OOV Discount (OD). The chances to be OOVs vary for words with different counts of characters. We can directly attack this problem by adding features OD_i that reward or punish OOV words which contains with i characters, or $OD_{i,j}$ for OOVs contains with i to j characters. 4 OD features are used in this paper: 1, 2, 3 and 4+. For example, OD_3 counts 1 when the word *tao-fei-ke* is an OOV.

Method	Train	#Rule	Test	TFs	MT04	MT05	Speed
Separate	ICT	151M	ICT	×	34.82	33.06	2.48
	SF	148M	SF	×	35.29	33.22	2.55
	ME	141M	ME	×	33.71	30.91	2.34
	All	219M	Lattice	×	35.79	33.95	3.83
				√	35.85	33.76	6.79
Joint	ICT	151M	Character	√	36.92	34.69	17.66
	SF	148M			37.02	34.56	17.37
	ME	141M			36.78	34.17	17.23
	All	219M			**37.25****	**34.88****	17.52

Table 2: Comparison of Separate and Joint methods in terms of BLEU and speed (second per sentence). Columns *Train* and *Test* represents the tokenization methods for training and testing respectively. Column *TFs* stands for whether the 8 tokenization features is used (√) or not (×). *ICT*, *SF* and *ME* are segmenter names for preprocessing. *All* means combined corpus processed by the three segmenters. Lattice represent the system implemented as Dyer et al., (2008). ** means significantly (Koehn, 2004) better than Lattice ($p < 0.01$).

4 Experiments

In this section, we try to answer the following questions:

1. Does the joint method outperform conventional methods that separate tokenization from decoding. (Section 4.1)

2. How about the tokenization performance of the joint decoder? (Section 4.2)

4.1 Translation Evaluation

We use the SCFG model (Chiang, 2007) for our experiments. We firstly work on the Chinese-English translation task. The bilingual training data contains 1.5M sentence pairs coming from LDC data.[1] The monolingual data for training English language model includes Xinhua portion of the GIGAWORD corpus, which contains 238M English words. We use the NIST evaluation sets of 2002 (MT02) as our development data set, and sets of 2004(MT04) and 2005(MT05) as test sets. We use the corpus derived from the People's Daily (Renmin Ribao) in Feb. to Jun. 1998 containing 6M words for training LM and ME tokenization models.

Translation Part. We used GIZA++ (Och and Ney, 2003) to perform word alignment in both directions, and grow-diag-final-and (Koehn et al., 2003) to generate symmetric word alignment. We extracted the SCFG rules as describing in Chiang (2007). The language model were trained by the SRILM toolkit (Stolcke, 2002).[2] Case insensitive NIST BLEU (Papineni et al., 2002) was used to measure translation performance.

Tokenization Part. We used the toolkit implemented by Zhang (2004) to train the ME model. Three Chinese word segmenters were used for comparing: ICTCLAS (*ICT*) developed by institute of Computing Technology Chinese Academy of Sciences (Zhang et al., 2003); *SF* developed at Stanford University (Huihsin et al., 2005) and *ME* which exploits the ME model described in section (3.2).

4.1.1 Joint Vs. Separate

We compared our joint tokenization and translation with the conventional separate methods. The input of separate tokenization and translation can either be a single segmentation or a lattice. The lattice combines the 1-best segmentations of segmenters. Same as Dyer et al., (2008), we also extracted rules from a combined bilingual corpus which contains three copies from different segmenters. We refer to this version of rules as *All*.

Table 2 shows the result.[3] Using *all* rule table, our joint method significantly outperforms the best single system *SF* by $+1.96$ and $+1.66$ points on MT04 and MT05 respectively, and also outperforms the lattice-based system by $+1.46$ and $+0.93$ points. However, the 8 tokenization features have small impact on the lattice system, probably because the tokenization space limited

[1] including LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08 and LDC2005T06

[2] The calculation of LM probabilities for OOVs is done by the SRILM without special treatment by ourself.

[3] The weights are retrained for different test conditions, so do the experiments in other sections.

ME	LM	WC	OCC	OD	MT05
×	×	×	×	×	24.97
√	×	×	×	×	25.30
×	√	×	×	×	24.70
×	×	√	×	×	24.84
×	×	×	√	×	25.51
×	×	×	×	√	25.34
×	√	√	×	×	25.74
√	√	√	√	√	26.37

Table 3: Effect of tokenization features on Chinese-English translation task. "√" denotes using a tokenization feature while "×" denotes that it is inactive.

Method	BLEU	#Word	Grau	#OOV
ICT	33.06	30,602	1.65	644
SF	33.22	30,119	1.68	882
ME	30.91	29,717	1.70	1,614
Lattice	33.95	30,315	1.66	494
Joint$_{ICT}$	34.69	29,723	1.70	996
Joint$_{SF}$	34.56	29,839	1.69	972
Joint$_{ME}$	34.17	29,771	1.70	1,062
Joint$_{All}$	34.88	29,644	1.70	883

Table 4: Granularity (Grau, counts of character per word) and counts of OOV words of different methods on MT05. The subscript of joint means the type of rule table.

by lattice has been created from good tokenization. Not surprisingly, our decoding method is about 2.6 times slower than lattice method with tokenization features, since the joint decoder takes character sequences as input, which is about 1.7 times longer than the corresponding word sequences tokenized by segmenters. (Section 4.1.4).

The number of extracted rules with different segment methods are quite close, while the *All* version contains about 45% more rules than the single systems. With the same rule table, our joint method improves the performance over separate method up to $+3.03$ and $+3.26$ points (*ME*). Interestingly, comparing with the separate method, the tokenization of training data has smaller effect on joint method. The BLEU scores of MT04 and MT05 fluctuate about 0.5 and 0.7 points when applying the joint method, while the difference of separate method is up to 2 and 3 points respectively. It shows that the joint method is more robust to segmentation performance.

4.1.2 Effect of Tokenization Model

We also investigated the effect of tokenization features on translation. In order to reduce the time for tuning weights and decoding, we extracted rules from the FBIS part of the bilingual corpus, and trained a 4-gram English language model on the English side of FBIS.

Table 3 shows the result. Only using the 8 translation features, our system achieves a BLEU score of 24.97. By activating all tokenization features, the joint decoder obtains an absolute improvement by 1.4 BLEU points. When only adding one single tokenization feature, the *LM* and *WC* fail to show improvement, which may result from their bias to short or long tokenizations. However, these two features have complementary advantages and collaborate well when using them together (line 8). The OCC and OD features also contribute improvements which reflects the fact that handling the generation of OOV is important for the joint model.

4.1.3 Considering All Tokenizations?

In order to explain the necessary of considering all potential tokenizations, we compare the performances of whether to tokenize a span as a single word or not as illustrated in section 3.3. When only tokenizing by the extracted rules, we obtain 34.37 BLEU on MT05, which is about 0.5 points lower than considering all tokenizations shown in Table 2. This indicates that spuriously limitation of the tokenization space may degenerate translation performance.

4.1.4 Results Analysis

To better understand why the joint method can improve the translation quality, this section shows some details of the results on the MT05 data set.

Table 4 shows the granularity and OOV word counts of different configurations. The lattice method reduces the OOV words quite a lot which is 23% and 70% comparing with ICT and ME. In contrast, the joint method gain an absolute improvement even thought the OOV count do not decrease. It seems the lattice method prefers to translate more characters (since smaller granularity and less OOVs), while our method is inclined to maintain integrity of words (since larger granularity and more OOVs). This also explains the difficulty of deciding optimal tokenization for translation before decoding.

There are some named entities or idioms that

Method	Type	F_1	Time
Monolingual	ICT	97.47	0.010
	SF	97.48	0.007
	ME	95.53	0.008
Joint	ICT	97.68	9.382
	SF	97.68	10.454
	ME	97.60	10.451
	All	97.70	9.248

Table 5: Comparison of segmentation performance in terms of F_1 score and speed (second per sentence). *Type* column means the segmenter for monolingual method, while represents the rule tables used by joint method.

are split into smaller granularity by the segmenters. For example:"史 东" which is an English name "Stone" or "豆-蔻 年-华" which means "teenage". Although the separate method is possible to translate them using smaller granularity, the translation results are in fact wrong. In contrast, the joint method tokenizes them as entire OOV words, however, it may result a better translation for the whole sentence.

We also count the overlap of the segments used by the $Joint_{All}$ system towards the single segmentation systems. The tokenization result of $Joint_{All}$ contains $29,644$ words, and shares $28,159$, $27,772$ and $27,407$ words with ICT, SF and ME respectively. And 46 unique words appear only in the joint method, where most of them are named entity.

4.2 Chinese Word Segmentation Evaluation

We also test the tokenization performance of our model on Chinese word segmentation task. We randomly selected 3k sentences from the corpus of People's Daily in Jan. 1998. 1k sentences were used for tuning weights, while the other 2k sentences were for testing. We use MERT (Och, 2003) to tune the weights by minimizing the error measured by F_1 score.

As shown in Table 5, with all features activated, our joint decoder achieves an F_1 score of 97.70 which reduces the tokenization error comparing with the best single segmenter *ICT* by 8.7%. Similar to the translation performance evaluation, our joint decoder outperforms the best segmenter with any version of rule tables.

Feature	F_1
TFs	97.37
TFs + RS	97.65
TFs + LM	97.67
TFs + RS + LM	97.62
All	97.70

Table 6: Effect of the target side information on Chinese word segmentation. *TFs* stands for the 8 tokenization features. *All* represents all the 16 features.

4.2.1 Effect of Target Side Information

We compared the effect of the 4 Rule Scores (RS), target side Language Model (LM) on tokenization. Table 6 shows the effect on Chinese word segmentation. When only use tokenization features, our joint decoder achieves an F_1 score of 97.37. Only integrating language model or rule scores, the joint decoder achieves an absolute improvement of 0.3 point in F_1 score, which reduces the error rate by 11.4%. However, when combining them together, the F_1 score deduces slightly, which may result from the weight tuning. Using all feature, the performance comes to 97.70. Overall, our experiment shows that the target side information can improve the source side tokenization under a supervised way, and outperform state-of-the-art systems.

4.2.2 Best Tokenization = Best Translation?

Previous works (Zhang et al., 2008; Chang et al., 2008) have shown that preprocessing the input string for decoder by better segmenters do not always improve the translation quality, we reverify this by testing whether the joint decoder produces good tokenization and good translation at the same time. To answer the question, we used the feature weights optimized by maximizing BLEU for tokenization and used the weights optimized by maximizing F_1 for translation. We test BLEU on MT05 and F_1 score on the test data used in segmentation evaluation experiments. By tuning weights regarding to BLEU (the configuration for Joint_{All} in table 2), our decoder achieves a BLEU score of 34.88 and an F_1 score of 92.49. Similarly, maximizing F_1 (the configuration for the last line in table 6) leads to a much lower BLEU of 27.43, although the F_1 is up to 97.70. This suggests that better tokenization may not always lead to better translations and vice versa

Rule	#Rule	Method	Test	Time
Morph	46M	Separate	21.61	4.12
Refined	55M		21.21	4.63
All	74M	Joint	21.93*	5.10

Table 7: Comparison of Separate and Joint method in terms of BLEU score and decoding speed (second per sentence) on Korean-Chinese translation task.

even by the joint decoding. This also indicates the hard of artificially defining the best tokenization for translation.

4.3 Korean-Chinese Translation

We also test our model on a quite different task: Korean-Chinese. Korean is an agglutinative language, which comes from different language family comparing with Chinese.

We used a newswire corpus containing 256k sentence pairs as training data. The development and test data set contain 1K sentence each with one single reference. We used the target side of training set for language model training. The Korean part of these data were tokenized into morpheme sequence as atomic unit for our experiments.

We compared three methods. First is directly use morpheme sequence (Morph). The second one is refined data (Refined), where we use selective morphological segmentation (Oflazer, 2008) for combining morpheme together on the training data. Since the selective method needs alignment information which is unavailable in the decoding, the test data is still of morpheme sequence. These two methods still used traditional decoding method. The third one extracting rules from combined (All) data of methods 1 and 2, and using joint decoder to exploit the different granularity of rules.

Table 7 shows the result. Since there is no gold standard data for tokenization, we do not use ME and LM tokenization features here. However, our joint method can still significantly ($p < 0.05$) improve the performance by about +0.3 points. This also reflects the importance of optimizing granularity for morphological complex languages.

5 Related Work

Methods have been proposed to optimize tokenization for word alignment. For example, word alignment can be simplified by packing (Ma et al., 2007) several consecutive words together. Word alignment and tokenization can also be optimized by maximizing the likelihood of bilingual corpus (Chung and Gildea, 2009; Xu et al., 2008). In fact, these work are orthogonal to our joint method, since they focus on training step while we are concerned of decoding. We believe we can further the performance by combining these two kinds of work.

Our work also has connections to multilingual tokenization (Snyder and Barzilay, 2008). While they have verified that tokenization can be improved by multilingual learning, our work shows that we can also improve tokenization by collaborating with translation task in a supervised way.

More recently, Liu and Liu (2010) also shows the effect of joint method. They integrate parsing and translation into a single step and improve the performance of translation significantly.

6 Conclusion

We have presented a novel method for joint tokenization and translation which directly combines the tokenization model into the decoding phase. Allowing tokenization and translation to collaborate with each other, tokenization can be optimized for translation, while translation also makes contribution to tokenization performance under a supervised way. We believe that our approach can be applied to other string-based model such as phrase-based model (Koehn et al., 2003), string-to-tree model (Galley et al., 2006) and string-to-dependency model (Shen et al., 2008).

Acknowledgement

The authors were supported by SK Telecom C&I Business, and National Natural Science Foundation of China, Contracts 60736014 and 60903138. We thank the anonymous reviewers for their insightful comments. We are also grateful to Wenbin Jiang, Zhiyang Wang and Zongcheng Ji for their helpful feedback.

References

Chang, Pi-Chuan, Michel Galley, and Christopher D. Manning. 2008. Optimizing Chinese word segmentation for machine translation performance. In *the Third Workshop on SMT*.

Chiang, David. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Chung, Tagyoung and Daniel Gildea. 2009. Unsupervised tokenization for machine translation. In *Proc. EMNLP 2009*.

Dyer, Christopher, Smaranda Muresan, and Philip Resnik. 2008. Generalizing word lattice translation. In *Proc. ACL 2008*.

Dyer, Chris. 2009. Using a maximum entropy model to build segmentation lattices for mt. In *Proc. NAACL 2009*.

Galley, Michel, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proc. ACL 2006*.

Huihsin, Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A conditional random field word segmenter. In *Fourth SIGHAN Workshop*.

Jiang, Wenbin, Liang Huang, Qun Liu, and Yajuan Lü. 2008. A cascaded linear model for joint chinese word segmentation and part-of-speech tagging. In *Proc. ACL 2008*.

Koehn, Philipp, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. HLT-NAACL 2003*.

Koehn, Philipp. 2004. Statistical significance tests for machine translation evaluation. In *Proc. EMNLP 2004*.

Liu, Yang and Qun Liu. 2010. Joint parsing and translation. In *Proc. Coling 2010*.

Liu, Yang, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proc. ACL 2006*.

Ma, Yanjun, Nicolas Stroppa, and Andy Way. 2007. Bootstrapping word alignment via word packing. In *Proc. ACL 2007*.

Mi, Haitao, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proc. of ACL 2008*.

Ng, Hwee Tou and Jin Kiat Low. 2004. Chinese part-of-speech tagging: One-at-a-time or all-at-once? word-based or character-based? In *Proc. EMNLP 2004*.

Och, Franz J. and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proc. ACL 2002*.

Och, Franz Josef and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Och, Franz Josef. 2003. Minimum error rate training in statistical machine translation. In *Proc. ACL 2003*.

Oflazer, Kemal. 2008. Statistical machine translation into a morphologically complex language. In *Proc. CICL 2008*.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. ACL 2002*.

Shen, Libin, Xu Jinxi, and Weischedel Ralph. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proc. ACL 2008*.

Snyder, Benjamin and Regina Barzilay. 2008. Unsupervised multilingual learning for morphological segmentation. In *Proc. ACL 2008*.

Stolcke, Andreas. 2002. Srilm – an extensible language modeling toolkit.

Xu, Jia, Evgeny Matusov, Richard Zens, and Hermann Ney. 2005. Integrated chinese word segmentation in statistical machine translation. In *Proc. IWSLT2005*.

Xu, Jia, Jianfeng Gao, Kristina Toutanova, and Hermann Ney. 2008. Bayesian semi-supervised chinese word segmentation for statistical machine translation. In *Proc. Coling 2008*.

Xue, Nianwen and Libin Shen. 2003. Chinese word segmentation as LMR tagging. In *SIGHAN Workshop*.

Zhang, Hua-Ping, Hong-Kui Yu, De-Yi Xiong, and Qun Liu. 2003. Hhmm-based chinese lexical analyzer ictclas. In *the Second SIGHAN Workshop*.

Zhang, Ruiqiang, Keiji Yasuda, and Eiichiro Sumita. 2008. Improved statistical machine translation by multiple Chinese word segmentation. In *the Third Workshop on SMT*.

Zhang, Le. 2004. Maximum entropy modeling toolkit for python and c++.

Build Chinese Emotion Lexicons Using A Graph-based Algorithm and Multiple Resources

Ge Xu, Xinfan Meng, Houfeng Wang
Key Laboratory of Computational Linguistics (Peking University), Ministry of Education
Institute of Computational Linguistics, Peking University
{xuge, mxf, wanghf}@pku.edu.cn

Abstract

For sentiment analysis, lexicons play an important role in many related tasks. In this paper, aiming to build Chinese emotion lexicons for public use, we adopted a graph-based algorithm which ranks words according to a few seed emotion words. The ranking algorithm exploits the similarity between words, and uses multiple similarity metrics which can be derived from dictionaries, unlabeled corpora or heuristic rules. To evaluate the adopted algorithm and resources, two independent judges were asked to label the top words of ranking list.

It is observed that noise is almost unavoidable due to imprecise similarity metrics between words. So, to guarantee the quality of emotion lexicons, we use an iterative feedback to combine manual labeling and the automatic ranking algorithm above. We also compared our newly constructed Chinese emotion lexicons (happiness, anger, sadness, fear and surprise) with existing counterparts, and related analysis is offered.

1 Introduction

Emotion lexicons have a great impact on the results of related tasks. With high-quality emotion lexicons, systems using simple methods can achieve competitive performance. However, to manually build an emotion lexicon is time-consuming. Many research works in building lexicons use automatic methods to assist the building procedure. Such works commonly rank words by the similarities to a set of seed words, then those words with high ranking scores are more likely to be added to the final lexicons or used as additional seed words.

For Chinese, emotion lexicons are scarce resources. We can get a small set of emotion words from semantic dictionary (such as CCD, HowNet, synonym dictionaries) or directly from related papers (Xu and Tao, 2003) (Chen et al. , 2009), but it is often not sufficient for practical systems. Xu et al. (2008) constructed a large-scale emotion ontology dictionary, but it is not publicly available yet.

In this paper, we adopted a graph-based algorithm to automatically rank words according to a few seed words. Similarity between words can be utilized and multiple resources are used to boost performance. Combining manual labeling with automatic ranking through an iterative feedback framework, we can produce high-quality emotion lexicons. Our experiments focused on Chinese, but the method is applicable to any other language as long as suitable resources exist.

The remainder of this paper is organized as follows. In Section 2, related works are introduced. In Section 3, we describe a graph-based algorithm and how to incorporate multiple resources. Section 4 gives the details of applying the algorithm on five emotions and shows how to evaluate the results. Section 5 focuses on how to build and evaluate emotion lexicons, linguistic consideration and instruction for identifying emotions are also included. Finally, conclusion is made in Section 6.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1209–1217,
Beijing, August 2010

2 Related work

Riloff and Shepherd (1997) presented a corpus-based method that can be used to build semantic lexicons for specific categories. The input to the system is a small set of seed words for a category and a representative text corpus. The output is a ranked list of words that are associated with the category. An approach proposed by (Turney, 2002) for the construction of polarity started with a few positive and negative seeds, then used a similarity method (pointwise mutual information) to grow this seed list from web corpus. Our experiments are similar with these works, but we use a different ranking method and incorporate multiple resources. To perform rating inference on reviews, Goldberg and Zhu (2006) created a graph on both labeled and unlabeled reviews, and then solved an optimization problem to obtain a smooth rating function over the whole graph. Rao and Ravichandran (2009) used three semi-supervised methods in polarity lexicon induction based on WordNet, and compared them with corpus-based methods. Encouraging results show methods using similarity between words can improve the performance. Wan and Xiao (2009) presented a method to use two types of similarity between sentences for document summarization, namely similarity within a document and similarity between documents. The ranking method in our paper is similar to the ones used in above three papers, which fully exploit the relationship between any pair of sample points (both labeled and unlabeled). When only limited labeled data are available, such method achieves significantly better predictive accuracy over other methods that ignore the unlabeled examples during training.

Xu et al. (2008) at first formed a taxonomy for emotions, under which an affective lexicon ontology exploiting various resources was constructed. The framework of ontology is filled by the combination of manual classification and automatic methods. To our best knowledge, this affective lexicon ontology is the largest Chinese emotion-oriented dictionary.

3 Our method

3.1 A graph-based algorithm

For our experiments, we chose the graph-based algorithm in (Zhou et al. , 2004) which is transductive learning and formulated as follows:

Given a point set $\chi = \{x_1, ..., x_l, x_{l+1}, ..., x_n\}$, the first l points $x_i (i \leq l)$ are labeled and the remaining points $x_u (l+1 \leq u \leq n)$ unlabeled. The goal is to rank the unlabeled points.

Let F denotes an n-dimensional vector whose elements correspond to ranking scores on the data set χ. Define another n-dimensional vector Y with $Y_i = 1$ if x_i is labeled and $Y_i = 0$ otherwise. Y denotes the initial label assignment.

The iterative algorithm is shown in the following:

Algorithm 1 A graph-based algorithm

1. Construct the weight matrix W and set W_{ii} to zero to avoid self-reinforcement. W is domain-dependent.
2. Construct the similarity matrix $S = D^{1/2} W D^{1/2}$ using symmetric normalization. D is a diagonal matrix with $D_{ii} = \Sigma_j W_{ij}$.
3. Iterate $F(t+1) = \alpha S F(t) + (1-\alpha) Y$ until convergence, where α is a parameter in (0, 1), and $F(0) = Y$. We clamp labeled points to 1 after each iteration.
4. Let F^* denote $F(t)$ when the iteration converges.

In our experiments, labeled points are seed emotion words, S_{ij} denotes the similarity between *i*th word and *j*th word. In an iteration, each word absorbs label information from other words. More similar two words are, more influence they have on each other. The label information (initially from seed emotion words) will propagate along S. The final output F^* contains ranking scores for all words, and a score indicates how similar the corresponding word is to the seed emotion words.

The implementation of the iterative algorithm is theoretically simple, which only involves basic matrix operation. Compared with methods which do not exploit the relationship between samples, experiments showing advantages of graph-based learning methods can be found

in (Rao and Ravichandran, 2009),(Goldberg and Zhu, 2006),(Tong et al. , 2005),(Wan and Xiao, 2009),(Zhu and Ghahramani, 2002) etc. When labeled data are scarce, such graph-based transductive learning methods are especially useful.

3.2 Incorporate multiple resources

For building the emotion lexicons, we are faced with lots of resources, such as semantic dictionaries, labeled or unlabeled corpora, and some linguistic experiences which can be presented as heuristic rules. Naturally we want to use these resources together, thus boosting the final performance. In graph-base setting, such resources can be used to construct the emotion-oriented similarity between words, and similarities will be represented by matrices.

The schemes to fuse similarity matrices are presented in (Sindhwani et al. , 2005), (Zhou and Burges, 2007), (Wan and Xiao, 2009) and (Tong et al. , 2005) etc. In our paper, not aiming at comparing different fusion schemes, we used a linear fusion scheme to fuse different similarities matrices from different resources. The scheme is actually a convex combination of matrices, with weights specified empirically.

The fusion of different similarity matrices falls in the domain of multi-view learning. A well-known multi-view learning method is Co-Training, which uses two views (two resources) to train two interactive classifiers (Blum and Mitchell, 1998). Since we focus on building emotion lexicons using multiple resources (multiple views), those who want to see the advantages of multi-view learning over learning with one view can refer to (Blum and Mitchell, 1998), (Sindhwani et al. , 2005), (Zhou and Burges, 2007), (Wan and Xiao, 2009) and (Tong et al. , 2005) etc.

4 Experiments

We use the method in section 3 to rank for each emotion with a few seed emotion words. Once we implement the ranking algorithm 1, the main work resides in constructing similarity matrices, which are highly domain-dependent.

4.1 Construct similarity matrices

Here, we introduce how to construct four similarity matrices used in building emotion lexicons. Three of them are based on cooccurrence of words; the fourth matrix is from a heuristic rule.

We use ictclas3.0[1] to perform word segmentation and POS tagging.

In our experiments, the number of words involved in ranking is 93506[2], so theoretically, the matrices are 93506×93506. If the similarity between any pair of words is considered, the computation becomes impractical in both time and space cost. So we require that each word has at most 500 nearest neighbors.

Four matrices are constructed as follows:

4.1.1 Similarity based on a unlabeled corpus

The unlabeled corpus used is People's Daily[3](人民日报1997∼2004). After word segmentation and POS tagging, we chose three POS's (i,a,l)[4]. The nouns were not included to limit the scale of word space. We set the cooccurrence window to a sentence, and removed the duplicate occurrences of words. Any pair of words in a sentence will contribute a unit weight to the edge which connects the pair of words.

4.1.2 Similarity based on a synonym dictionary

We used the Chinese synonym dictionary (哈工大同义词词林扩展版[5]) for this matrix. In this dictionary, the words in a synonym set are presented in one line and separated by spaces, so there is no need to perform word segmentation and POS tagging. Any pair of words in one line will contribute a unit weight to the edge which connects the pair of words.

4.1.3 Similarity based on a semantic dictionary

We used The Contemporary Chinese Dictionary (现代汉语词典) to construct the third simi-

[1] downloaded from http://www.ictclas.org/

[2] Words are selected after word segmentation and POS tagging, see section 4.1.1∼4.1.3 for selection of words in details.

[3] http://icl.pku.edu.cn/

[4] i=Chinese idiom, a=adjective, l=Chinese phrase

[5] http://ir.hit.edu.cn/

larity matrix. Since word segmentation may segment the entries of the dictionary, we extracted all the entries in the dictionary and store them in a file whose words ictclas3.0 was required not to segment. Furthermore, for an entry in the dictionary, the example sentences or phrases appearing in its gloss may contain many irrelevant words in terms of emotions, so they were removed from the gloss.

After word segmentation and POS tagging[6], we set the cooccurrence window to one line (an entry and its gloss without example sentences or phrases), and removed the duplicate occurrences of words. An entry and any word in the modified gloss will contribute a unit weight to the edge which connects the pair of words. This constructing was a bit different, since we did not consider the similarity between words in modified gloss.

4.1.4 similarity based on a heuristic rule

In Chinese, a word is composed of one or several Chinese characters. A Chinese character is normally by itself an independent semantic unit, so the similarity between two words can be inferred from the character(s) that they share. For example, the Chinese word 欣 (happy) appears in the word 欣然 (readily). Since 欣然 and 欣 share one Chinese character, they are regarded as similar. Naturally, the larger the proportion that two words share, the more similar they are. In this way, the fourth weighted matrix was formed. To avoid incurring noises, we exclude the cases where one Chinese character is shared, with the exception that the Chinese character itself is one of the two Chinese words.

4.1.5 Fusion of four similarity matrices

After processing all the lines (or sentences), the weighted matrices are normalized as in algorithm 1, then four similarity matrices are linearly fused with equal weights (1/4 for each matrix).

4.2 Select seed emotion words

In our experiments, we chose emotions of *happiness, sadness, anger, fear and surprise* which are widely accepted as basic emotions[7]. Empirically, we assigned each emotion with seed words given in Table 1.

Emotion	**Seed words**
喜(happiness)	高兴, 愉快, 欢乐, 喜悦, 兴高采烈, 欢畅, 开心
怒(anger)	愤怒, 不满, 恼火, 生气, 愤恨, 恼怒, 愤懑, 震怒, 悲愤, 窝火, 痛恨, 恨之入骨, 义愤填膺, 怒气冲天
哀(sadness)	悲伤, 沮丧, 痛苦, 伤心, 难过, 悲哀, 难受, 消沉, 灰心丧气, 悲戚, 闷闷不乐, 哀伤, 悲愤, 悲切, 悲痛欲绝, 欲哭无泪
惧(fear)	恐惧, 惧怕, 担心, 提心吊胆, 害怕, 惊恐, 疑惧, 畏惧, 不寒而栗, 望而生畏
惊(surprise)	惊讶, 大吃一惊 ,震惊, 惊恐, 惊异, 惊骇, 惊, 出乎意料, 惊喜, 惊叹

Table 1: Seed emotion words

4.3 Evaluation of our method

We obtained five ranking lists of words using the method in section 3. Following the work of (Riloff and Shepherd, 1997), we adopted the following evaluation setting.

To evaluate the quality of emotion ranking lists, each list was manually rated by two persons independently. For each emotion, we selected the top 200 words of each ranking list and presented them to judges. We presented the words in random order so that the judges had no idea how our system had ranked the words. The judges were asked to rate each word on a scale from 1 to 5 indicating how strongly it was associated with an emotion, 0 indicating no association. We allowed the judges to assign -1 to a word if they did not know what it meant. For the words rated as -1, we manually assigned ratings that we thought were appropriate.

The results of judges are shown in figures 1-5. In these figures, horizontal axes are the number of reviewed words in ranking lists and vertical axes are number of emotion words found (with 5 different strength). The curve labeled as $> x$ means that it counts the number of words which are rated

[6]since we do not segment entries in this dictionary, all POS's are possible

[7]Guidelines for identifying emotions is in section 5, before that, we understand emotions through common sense.

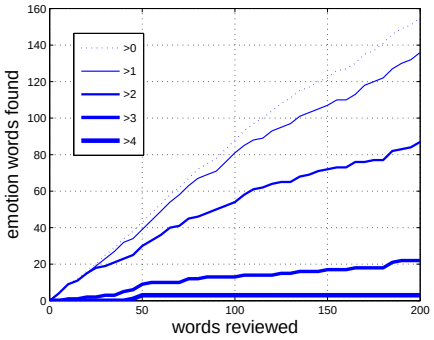


Figure 1: happiness

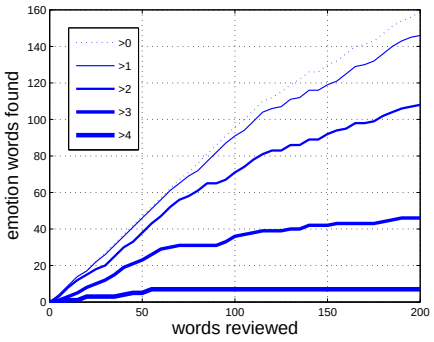


Figure 2: anger

greater than x by either judge.

Curves ($> 0, > 1, > 2$) display positive slopes even at the end of the 200 words, which implies that more emotion words would occur if more than 200 words are reviewed. By comparison, curves ($> 3, > 4$) tend to be flat when they are close to the right side, which means the cost of identifying high-quality emotion words will increase greatly as one checks along the ranking list in descendent order.

It is observed that words which both judges assign 5 are few. In *surprise* emotion, the number is even 0. Such results may reflect that emotion is harder to identify compared with topical categories in (Riloff and Shepherd, 1997).

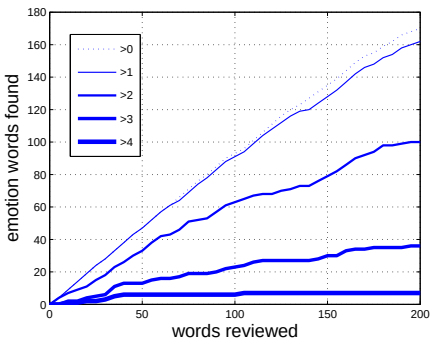


Figure 3: sadness

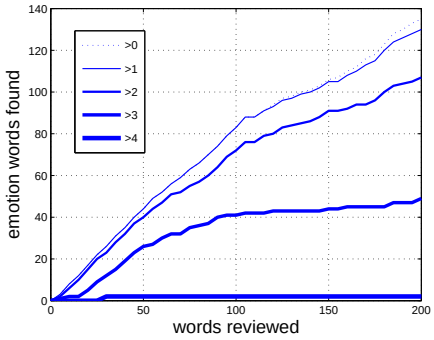


Figure 4: fear

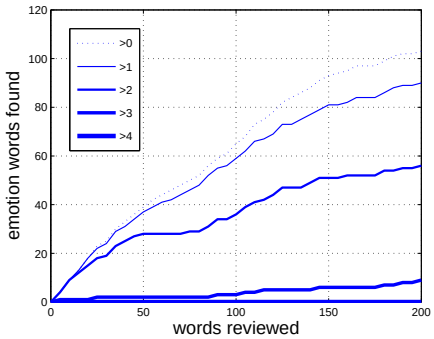


Figure 5: surprise

From the semantic dictionary, our method found many low-frequency emotion words such as 忭 (pleasant, glad), 蘧然 (surprise and happy), 忉怛 (sad), or those used in Chinese dialects such as 毛咕 (fear), 挂气 (angry). Such emotion words are necessary for comprehensive emotion lexicons.

Because more POS's than adjectives and verbs are included in our experiments, some emotion words such as the noun 冷门 (unexpected winner),and the adverb 竟然 (to one's surprise) are also spotted, which to some extent implies the generality of our method.

5 Construct emotion lexicons

The above section introduced a method to rank words with a few seed emotion words. However, to build emotion lexicons requires that we manually remove the noises incurred by the automatic ranking method. Accordingly, guidelines for identifying emotions are needed, and also some linguistic consideration in identifying emoting words should be given.

5.1 An iterative feedback to denoise

In our experiments, we observed that noises incurred by similarity matrices are almost unavoidable. For example, in the unlabeled corpus, 国事访问 (state visits) always co-occurred with 高兴 (happy) or 愉快 (happy), so in happiness emotion, 国事访问 acquired a high ranking position (174th); in terms of the heuristic rule, 意料 (expected) shares two Chinese characters with 出乎意料 (unexpected, surprised), however they have opposite meaning because 出乎 (exceed, beyond) is a negative word. 意料 unfavorably ranked high (88th) in surprise emotion; from the semantic dictionary, the gloss of 年画 (Chinese Spring Festival pictures) contains 欢乐 (happy), thus in happiness emotion, 年画 ranked high (158th).

So after each ranking of an emotion, in the descendent order of ranking scores, we manually revised some scores in about top 500. Several criteria (see 5.2 and 5.3) were given to guide if a word has a specified emotion. For those words surely bearing the specified emotion, we assigned 1 to them ,and left others unchanged. Seeing the words newly revised to be 1 as new seed emotion words, we run the ranking algorithm again. After such feedback was repeated 2∼3 times, we collected all the words labeled with 1 to form the final emotion lexicons. In (Zhou et al. , 2004), the author also suggested such *iterative feedback* to extend the query (seed) set and improve the ranking output. Commonly, the size of an emotion lexicon is small, so we do not have to check too many words.

The human revising procedure is sensitive to annotators' background. To improve the quality of the emotion lexicons, experts with linguistic or psychology background will help.

Furthermore, the ranking algorithm used in our paper is clearly sensitive to the initial seed words, but since we adopt an iterative feedback framework, the words not appearing in the initial set of seed words will show up in next iteration with high ranking scores. We also performed experiments which selected emotion seed words based on the Chinese synonym dictionary and the emotion words in (Chen et al. , 2009), similar results were found.

5.2 Guidelines for identifying emotions

The same as (Chen et al. , 2009), we used the definition that emotion is the felt awareness of bodily reactions to something perceived or thought. Also, we were highly influenced by the structure of the affective lexicon presented by (Ortony et al. , 1987), and used the Affective states and Affective-Behavioral conditions in the structure to identify emotion words in our paper[8].

With such guidelines, 胆小 (cowardice, relates more to external evaluation) is not an emotional word of fear. We also intentionally distinguish between emotions and expression of emotions. For example, 大笑 (laugh), 哈哈 (haw-haw) are seen as expression of happiness and 颤抖 (tremble) as of fear, but not as emotion words. In addition, we try to distinguish between an emotion and the cause of an emotion, see 5.3 for an example.

For each emotion, brief description is given as below[9]:

1. **Happiness**：the emotional reaction to something that is satisfying.
2. **Anger**：do not satisfy the current situation and have a desire to fight or change the situation. Often there exists a target for this emotion.
3. **Sadness**：an emotion characterized by feelings of disadvantage, loss, and helplessness. Sadness often leads to cry.
4. **Fear**：the emotional response to a perceived threat. Fear almost always relates to future events, such as worsening of a situation, or continuation of a situation that is unacceptable.
5. **Surprise**：the emotional reaction to something unexpected.

5.3 Linguistic consideration for identifying emotion words

If a word has multiple senses, we only consider its emotional one(s). For example, 生气 (as a verb, it means *be angry*, but means *vitality or spirits* as a noun) will appear in the emotion lexicon of anger.

[8] According to (Ortony et al. , 1987), *surprise* should not be seen as a basic emotion for it relates more to cognition. However, our paper focuses on the building of emotion lexicons, not the disputable issue of basic emotions

[9] we mainly referred to http://en.wikipedia.org/wiki

If one sense of a word is the combination of emotions, the word will appear in all related emotions.

We mainly consider four POS's, namely nouns, verbs, adjectives and adverb[10]. If a word has multiple POS's, we normally consider its POS with strongest emotion (Empirically, we think the emotion strength ranks in decedent order as following: adjectives, verbs, adverbs, nouns.). So we consider the verb of 恐惧 (fear) when it can be used as a noun and a verb in Chinese. The 生气 example above also applies here.

For each of four POS's, instruction for emotion identification is given as below:

Nouns: For example, 怒火 (rage, anger), 喜气 (joy or jubilation), 冷门 (an unexpected winner) are selected as emotion words. We distinguish between an emotion and the cause of an emotion. For example, calamity often leads to sadness, but does not directly contain the emotion of sadness. 冷门 appears in the surprise lexicon because we believe it contains surprise by itself.

Adverbs: The adverbs selected into emotion lexicons contain the emotions by themselves. For example, 竟然 (unexpectedly), 欣欣然 (cheerily), 气哼哼 (angrily), 蓦地 (unexpectedly), 伤心地 (sadly) etc.

Verbs: As in (Ortony et al. , 1987), Chinese emotion verbs also fall into at least two distinct classes, causatives and noncausatives. Both classes are included in our emotion lexicons. For example, 动肝火 (be angry), 担心 (fear) are noncausative verbs, while 激怒 (enrage), 震惊 (to make someone surprised) are causative ones. Probably due to the abundant usage of 令人/让人/使人 (to make someone) etc., causative emotion verbs are few compared to noncausative ones in Chinese.

Adjective：Quite a lot of emotion words fall in this POS, since adjectives are the natural expression of internal states of humans. For example, 高兴 (happy), 惊讶 (surprised), 愤怒 (angry) etc.

For any word that it is hard to identify at first sight, we used a search tool[11] to retrieve sentences which contain the word, and then identify if the word is emotional or not by its usage in the sentences.

[10]For Chinese idioms, we only considered those used as these four POS's, omitted those used as a statement, such as 哀兵必胜 (an army burning with righteous indignation is bound to win)

[11]provided by Center for Chinese Linguistics of Peking University, http://ccl.pku.edu.cn

5.4 Comparison with existing Chinese emotion resources

诧、骇、惊、讶、矍、蘧、愕、遽、骇然、赫然、竟然、居然、蘧然、愕然、愕然、矍然、爆冷、爆冷门、不料、不意、不虞、诧异、吃惊、出乎意料、出乎意外、出乎预料、出冷门、出其不意、出人意料、出人意外、触目惊心、错愕、大吃一惊、大惊失色、大惊小怪、怪讶、骇怪、骇然、骇人听闻、骇异、好家伙、赫然、赫然而怒、黑马、惊诧、惊呆、惊服、惊骇、惊慌、惊慌失措、惊惶、惊惶失措、惊魂未定、惊悸、惊惧、惊恐、惊恐万状、惊奇、惊人、惊世骇俗、惊叹、惊悉、惊喜、惊喜交集、惊喜万分、惊吓、惊羡、惊讶、惊疑、惊异、惊厥、惊愕、竟然、竟是、竟至、竟自、居然、冷不丁、冷不防、冷孤丁、冷门、没成想、猛不防、猛孤丁地、纳罕、始料不及、始料未及、受宠若惊、受惊、谁料、谁知、突如其来、未料、闻所未闻、想不到、心惊、心惊胆颤、心惊胆战、讶异、一语惊人、意料之外、意外、意想不到、又惊又喜、震惊、蓦地

Table 2: The emotion lexicon of surprise

Under the guidelines for manually identifying emotion words, we finally constructed five Chinese emotion lexicons using the iterative feedback. The newly constructed emotion lexicons were also reported as resources together with our paper. The emotion lexicon of *surprise* is shown in Table 2. In this part, we compare our lexicons with the following counterparts, see Table 3.

Ours1 in the table is the final emotion lexicons, and Ours2 is the abridged version that excludes the words of single Chinese character and Chinese idioms.

Chinese Concept Dictionary (CCD) is a WordNet-like semantic lexicon(Liu et al. , 2003).

	喜	怒	哀	惧	惊
CCD nouns	22	27	38	46	10
(Xu and Tao, 2003)	45	12	28	21	12
(Chen et al. , 2009)	28	34	28	17	11
(Xu et al. , 2008)	609	187	362	182	47
Ours1	95	118	97	106	99
Ours2	52	77	72	57	65

Table 3: Compare various emotion lexicons

We only considered the noun network which is richly developed in CCD, as in other semantic dictionaries. For each emotion, we chose its synset as well as the synsets of its hypernym and hyponym(s). In fact, most of words in the emotion nouns extracted can be used as verbs or adjectives in Chinese. However, since CCD is not designed for emotion analysis, words which are expression of emotions such as 哭泣 (cry) or evaluation such as 胆小 (cowardice) were included.

Selecting nouns and verbs, Xu and Tao (2003) offered an emotion taxonomy of 390 emotion words. The taxonomy contains 24 classes of emotions and excludes Chinese idioms. By our inspection to the offered emotion words in this taxonomy, the authors tried to exclude expression of emotions, evaluation and cause of emotions from emotions, which is similar with our processing[12]. Ours2 is intentionally created to compare with this emotion taxonomy.

Based on (Xu and Tao, 2003), Chen et al. (2009) removed the words of single Chinese character; let two persons to judge if a word is an emotional one and only those agreed by the two persons were seen as emotion words. It is worth noting that Chen et al. (2009) merges 怒 (anger) and 烦 (fidget) in (Xu and Tao, 2003) to form the 怒 (anger) lexicon, thus 讨厌 (dislike) appears in anger lexicon. However, we believe 讨厌 (dislike) is different with 怒 (anger), and should be put into another emotion. Also, we distinguish between 恨 (hate) and 怒 (anger).

Xu et al. (2008) constructed a large-scale affective lexicon ontology. Given the example words in their paper, we found that the authors did not intentionally exclude the expression of emotions such as 面红耳赤 (literally, red face and ear), 笑眯眯 (literally, be smiling). Such criteria of identifying emotion words may partially account for the large size of their emotion resources.

[12]Xu and Tao (2003) included words such as 情愿/愿意 (be willing to), 留神 (be careful) in their happiness lexicon, which we think should not be classified into happiness.

6 Conclusion and future work

In this paper, aiming to build Chinese emotion lexicons, we adopt a graph-based algorithm and incorporate multiple resources to improve the quality of lexicons and save human labor. This is an initial attempt to build Chinese emotion lexicons, the quality of constructed emotion lexicons is far from perfect and is supposed to be improved step by step.

The method in this paper can be further extended to subjectivity/polarity classification and other non-sentimental tasks such as word similarity computing, and can be also adapted to other languages. The more resources we use, the more human cost can be saved and the higher the quality of built emotion lexicons is.

In the future work, we want to construct other emotion lexicons such as 好 (like, love), 恶 (dislike), 欲 (desire) etc. using the same method.

Acknowledgement This research is supported by National Natural Science Foundation of China (No.60973053, No.90920011)

References

A. Blum and T. Mitchell. 1998. Combining labeled and unlabeled data with co-training。 *In Proceedings of the 11th Annual Conference on Computational Learning Theory*, 92-100.

Ying Chen, Sophia Y. M. Lee, and Churen Huang. 2009. A Cognitive-based Annotation System for Emotion Computing. *Proceedings of the Third Linguistic Annotation Workshop (LAW III)*.

Andrew B. Goldberg, Xiaojin Zhu. 2006. Seeing stars when there aren't many stars: graph-based semi-supervised learning for sentiment categorization. *Proceedings of TextGraphs: the First Workshop on Graph Based Methods for Natural Language Processing on the First Workshop on Graph Based Methods for Natural Language Processing*.

Y. Liu and et al. 2003. The CCD Construction Model and Its Auxiliary Tool VACOL. *Applied Linguistics, 45(1):83-88*.

A. Ortony, G. L. Clore, and M. A. Foss. 1987. The referential structure of the affective lexicon. *Cognitive Science, 11, 341-364*.

Delip Rao and D. Ravichandran. 2009. Semisupervised polarity lexicon induction. *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, 675-682.

Ellen Riloff and Jessica Shepherd. 1997. A Corpus-Based Approach for Building Semantic Lexicons. *In Proceedings of the Second Conference on Empirical Methods in Natural Language Processing, pages 117-124*.

V. Sindhwani, P. Niyogi, and M. Belkin. 2005. A co-regularization approach to semisupervised learning with multiple views. *Proc. ICML Workshop on Learning with Multiple views*.

H. Tong, J. He, M. Li, C. Zhang, and W. Ma. 2005. Graph based multi-modality learning. *In Proceedings of the 13th Annual ACM international Conference on Multimedia. MULTIMEDIA '05. ACM*, New York, NY, 862-871.

Peter D. Turney. 2002. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. *ACL 2002*, 417-424.

Xiaojun Wan and Jianguo Xiao. 2009. Graph-Based Multi-Modality Learning for Topic-Focused MultiDocument Summarization. *IJCAI 2009*, 1586-1591.

Linhong Xu, Hongfei Lin, Yu Pan, Hui Ren and Jianmei Chen. 2008. Constructing the Afective Lexicon Ontology. *JOURNAL OF THE CHINA SOCIETY F0R SCIENTIFIC AND TECHNICAL INFORMATION Vo1.27 No.2, 180-185*.

X. Y. Xu, and J. H. Tao. 2003. The study of affective categorization in Chinese. *The 1st Chinese Conference on Affective Computing and Intelligent Interaction. Beijing, China*.

Hongbo Xu, Tianfang Yao, and Xuanjing Huang. 2009. The second Chinese Opinion Analysis Evaluation(in Chinese). *COAE 2009*.

D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Scholkopf. 2004. Learning with local and global consistency. *Advances in Neural Information Processing Systems 16. MIT Press, Cambridge, MA.* .

D. Zhou and C. J. C. Burges. 2007. Spectral clustering and transductive learning with multiple views. *Proceedings of the 24th international conference on Machine learning*.

X. Zhu and Z. Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation. *Technical Report CMUCALD02107*. CMU.

A Methodology for Automatic Identification of Nocuous Ambiguity

Hui Yang[1] **Anne de Roeck**[1] **Alistair Willis**[1] **Bashar Nuseibeh**[1, 2]
[1]Department of Computing, The Open University
[2]Lero, University of Limerick
{h.yang, a.deroeck, a.g.willis, b.nuseibeh}@open.ac.uk

Abstract

Nocuous ambiguity occurs when a linguistic expression is interpreted differently by different readers in a given context. We present an approach to automatically identify nocuous ambiguity that is likely to lead to misunderstandings among readers. Our model is built on a machine learning architecture. It learns from a set of heuristics each of which predicts a factor that may lead a reader to favor a particular interpretation. An ambiguity threshold indicates the extent to which ambiguity can be tolerated in the application domain. Collections of human judgments are used to train heuristics and set ambiguity thresholds, and for evaluation. We report results from applying the methodology to coordination and anaphora ambiguity. Results show that the method can identify nocuous ambiguity in text, and may be widened to cover further types of ambiguity. We discuss approaches to evaluation.

1 Introduction

Traditional accounts of ambiguity have generally assumed that each use of a linguistic expression has a unique intended interpretation in context, and attempted to develop a model to determine it (Nakov and Hearst, 2005; Brill and Resnik, 1994). However, disambiguation is not always appropriate or even desirable (Poesio and Artstein, 2008). Ambiguous text may be interpreted differently by different readers, with no consensus about which reading is the intended one. Attempting to assign a preferred interpretation may therefore be inappropriate. Misunderstandings among readers do occur and may have undesirable consequences. In requirements engineering processes, for example, this results in costly implementation errors (Boyd et al., 2005).

Nonetheless, most text does not lead to significant misinterpretation. Our research aims to establish a model that estimates how likely an ambiguity is to lead to misunderstandings. Our previous work on nocuous ambiguity (Chantree et al., 2006; Willis et al., 2008) cast ambiguity not as a property of a text, but as a property of text in relation to a set of stakeholders. We drew on human judgments - interpretations held by a group of readers of a text – to establish criteria for judging the presence of nocuous ambiguity. An ambiguity is *innocuous* if it is read in the same way by different people, and *nocuous* otherwise. The model was tested on co-ordination ambiguity only.

In this paper, we implement, refine and extend the model. We investigate two typical ambiguity types arising from coordination and anaphora. We extend the previous work (Willis et al., 2008) with additional heuristics, and refine the concept of ambiguity threshold. We experiment with alternative machine learning algorithms to find optimal ways of combining the output of the heuristics. Yang et al. (2010a) describes a complete implementation in a prototype tool running on full text. Here we present our experimental results, to illustrate and evaluate the extended methodology.

The rest of the paper is structured as follows. Section 2 introduces the methodology for automatic detection of nocuous ambiguity. Sections 3 and 4 provide details on how the model is applied to coordination and anaphora ambiguity. Experimental setup and results are reported in Section 5, and discussed in Section 6. Section 7 reports on related work. Conclusions and future work are found in Section 8.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1218–1226,
Beijing, August 2010

2 Methodology for Nocuous Ambiguity Identification

This section describes the main ideas underpinning our model of ambiguity. We distinguish between structural and interpretative aspects. The former captures the fact that text may have structure (i.e. syntax) which, in principle, permits multiple readings. These are relatively straightforward to identify from the linguistic constructs present in the text. The latter acknowledges that if text is interpreted in the same way by different readers, it has a low risk of being misunderstood. Modelling interpretive aspects requires access to human judgments about texts. Our approach has three elements, which we describe in turn: collection of human judgments; heuristics that model those judgments, and a machine learning component to train the heuristics.

Human judgments. We define an ambiguity as nocuous if it gives rise to diverging interpretations. Wasow et al. (2003) suggests that ambiguity is always a product of the meaning that people assign to language, and thus a subjective phenomenon. We capture individual interpretations of instances of ambiguity by surveying participants, asking them for their interpretation. We use this information to decide whether, given some ambiguity threshold, a particular instance is seen as innocuous or nocuous depending on the degree of dissent between judges.

A key concept in determining when ambiguity is nocuous is the *ambiguity threshold*. Different application areas may need to be more or less tolerant of ambiguity (Poesio and Artstein, 2008). For instance, requirements documents describing safety critical systems should seek to avoid misunderstandings between stakeholders. Other cases, such as cookbooks, could be less sensitive. Willis et al. (2008)'s general concept of ambiguity threshold sought to implement a flexible tolerance level to nocuous ambiguity. Given an instance of ambiguous text, and a set of judgments as to the correct interpretation, the *certainty* of an interpretation is the percentage of readers who assign that interpretation to the text. For example, in Table 1 below (sec. 3.1), the certainty of the two interpretations, HA and LA of expression (a) are 12/17=71% and 1/17=5.9% respectively. Here, an expression shows *nocuous ambiguity* if none of the possible interpretations have a certainty exceeding the chosen threshold. Later in this section, we will describe further experiments with alternative, finer grained approaches to setting and measuring thresholds, that affect the classifier's behaviour.

Heuristics. Heuristics capture factors that may favour specific interpretations. Each heuristic embodies a hypothesis, drawn from the literature, about a linguistic phenomenon signifying a preferred reading. Some use statistical information (e.g., word distribution information obtained from a generic corpus, the BNC[1], using the Sketch Engine[2]). Others flag the presence of surface features in the text, or draw on semantic or world knowledge extracted from linguistic resources like WordNet[3] or VerbNet[4].

Machine learning (ML). Individual heuristics have limited predictive power: their effectiveness lies in their ability to operate in concert. Importantly, the information they encapsulate may be interdependent. We harness this by using ML techniques to combine the outputs of individual heuristics. ML is an established method for recognizing complex patterns automatically, making intelligent decisions based on empirical data, and learning of complex and nonlinear relations between data points. Our model uses supervised learning ML techniques, deducing a function from training data, to classify instances of ambiguity into nocuous or innocuous cases. The classifier training data consists of pairs of input objects (i.e. vectors made up of heuristics scores) and desired outputs (i.e. the class labels determined by the distribution of human judgments as captured by thresholds). To select an appropriate ML algorithm for the nocuity classifier, we tested our datasets (described in later sections) on several algorithms in the WEKA[5] package (e.g., decision tree, J48, Naive Bayes, SVM, Logistic Regression, LogitBoost, etc.)

To train, and validate, a nocuity classifier for a particular form of ambiguity, we build a dataset of judgments, and select heuristics that model

[1] http://www.natcorp.ox.ac.uk/
[2] http://sketchengine.co.uk/
[3] http://wordnet.princeton.edu/
[4] http://verbs.colorado.edu/~mpalmer/projects/verbnet.html
[5] http://www.cs.waikato.ac.nz/~ml/index.html

the information underlying the human judgements about a preferred interpretation.

We validated the approach on two forms of ambiguity. Sections 3 and 4 discuss how the methodology is applied to forms of coordination and anaphoric ambiguity, and evaluate the performance of the final classifiers.

3 Automatic Identification of Nocuous Coordination Ambiguity

Our previous work on nocuous ambiguity has focused on coordination ambiguity: a common kind of structural ambiguity. A coordination structure connects two words, phrases, or clauses together via a coordination conjunction (e.g., '*and*', '*or*', etc) as in the following examples:

(1) *They support a typing system for architectural components and connectors.*

(2) *It might be rejected or flagged for further processing.*

In (1), the coordination construction '*architectural components and connectors*' consists of a **near conjunct** (NC) (i.e. '*components*'), a **far conjunct** (FC) (i.e. '*connectors*'), and the attached **modifier** (M) (i.e. '*architectural*'). This construction allows two bracketings corresponding to high modifier attachment (*[architectural [components and connectors]]*) or low modifier attachement (*[[architectural components] and connector]*). Our aim is to refine Chantree et al (2006) and Willis et al (2008), hence our focus is on the two phenomena they treated: modification in noun phrase coordination (as in (1)) and in verb phrase coordination (as in (2)).

We implemented the heuristics described in the earlier work, and introduced two further ones (local document collocation frequency, and semantic similarity). We used the Chantree et al (2006) dataset of human judgments, but employed the LogitBoost algorithm for implementing the nocuity classifier (rather than the Logistic Regression equation). The following subsections give more detail.

3.1 Building a dataset

Coordination instances. Our dataset was collected and described by Chantree et al. (2006). It contains 138 coordination instances gathered from a set of requirement documents. Noun compound conjunctions account for the majority (85.5%) of cases (118 instances). Nearly half of these arose as a result of noun modifiers, while there are 36 cases with adjective and 18 with preposition modifiers.

Human judgment collection. The coordination instances containing potential ambiguity were presented to a group of 17 computing professionals including academic staff or research students. For each instance, the judges were asked to select one of three options: high modifier attachment (HA), low modifier attachment (LA), or ambiguous (A). Table 1 shows the judgment count for two sample instances. In instance (a) in table 1, the *certainty* of HA is 12/17=71%, and the *certainty* of LA is 1/17=6%. Instance (b) was judged mainly to be ambiguous.

	Judgments		
	HA	LA	A
(a) *security* and *privacy* requirements	12	1	4
(b) electrical *characteristics* and *interface*	4	4	9

Table 1. Judgment count for the sample instances (HA=high attachment; LA=low attachment; and A=Ambiguous)

We set an ambiguity threshold, τ, to determine whether the distribution of interpretations is nocuous or innocuous with respect to that particular τ. If the *certainty* of neither interpretation, HA or LA, exceeds the threshold τ, we say this is an instance of *nocuous* coordination. Otherwise it is *innocuous*. Here, (a) displays nocuous ambiguity for τ>71%.

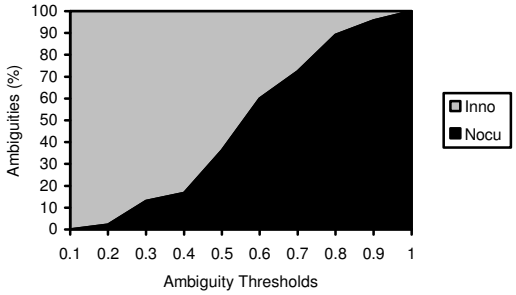


Figure 1. Proportions of interpretations at different ambiguity thresholds in the coordination instances

Figure 1 shows the systematic relationship between ambiguity threshold and the incidence of nocuous ambiguity in the dataset. Low thresholds can be satisfied with a very low certainty scores resulting in few instances being considered nocuous. At high thresholds, almost all instances are classified as nocuous unless the judges report a consensus interpretation.

3.2 Heuristics to predict Nocuity

Each heuristic tests a factor favouring a high or low modifier attachment (HA or LA). We implemented and extended Willis et al. (2008).

Coordination matching favours HA when the head words of near and far conjuncts are frequently found coordinated in a general corpus like BNC, suggesting they may form a single syntactic unit.

Distribution similarity measures how often two words are found in the same contexts. It favours HA where it detects a strong distributional similarity between the headwords of the two conjuncts, suggesting these form a syntactic unit (Kilgariff 2003).

Collocation frequency favours LA when the modifier is collocated much more frequently with the headword of the near conjunct than the far conjunct, in the document, or in the BNC.

Morphology favours HA when the conjunct headwords share a morphological marker (suffix) (Okumura and Muraki 1994).

Semantic similarity favours HA when the conjunct headwords display strong similarity in the taxonomic structure in WordNet[6].

3.3 Nocuity classification

To train, and test, the nocuity classifier, each ambiguity training/test instance is represented as an attribute-value vector, with the values set to the score of a particular heuristic. The class label of each instance (nocuous (Y) or innocuous (N) at a given ambiguity threshold) is determined by the *certainty* measure as discussed earlier. We selected the LogitBoost algorithm for building the classifier, because it outperformed other candidates on our training data than. To determine whether a test instance displays nocuity or not, we presented its feature vector to the classifier, and obtained a predicted class label (Y or N).

4 Automatic Identification of Nocuous Anaphora Ambiguity

An **anaphor** is an expression referring to an **antecedent**, usually a noun phrase (NP) found in the preceding text. A*naphora ambiguity* occurs when there are two or more candidate antecedents, as in example (3).

(3) *The procedure shall convert the 24 bit image to an 8 bit image, then display **it** in a dynamic window.*

In this case, both of the NPs, '*the 24 bit image*' and '*an 8 bit image*', are considered potential candidate antecedents of the anaphor '*it*'.

Anaphora ambiguity is difficult to handle due to contextual effects spread over several sentences. Our goal is to determine whether a case of anaphora ambiguity is nocuous or innocuous, automatically, by using our methodology.

4.1 The building of the Dataset

Anaphora instances. We collected 200 anaphora instances from requirements documents from RE@UTS website[7]. We are specifically concerned with 3rd person pronouns, which are widespread in requirements texts. The dataset contains different pronoun types. Nearly half the cases (48%) involve subject pronouns, although pronouns also occurred in objective and possessive positions (15% and 33%, respectively). Pronouns in prepositional phrases (e.g., '*under it')* are rarer (4% - only 8 instances).

Human judgment collection. The instances were presented to a group of 38 computing professionals (academic staff, research students, software developers). For each instance, the judges were asked to select the antecedent from the list of NP candidates. Each instance was judged by at least 13 people. Table 2 shows an example of judgment counts, where 12 out of 13 judges committed to '*supervisors*' as the antecedent of *'they'*, whereas 1 chose '*tasks*'.

1. Supervisors may only modify tasks **they** supervise to the agents they supervise.		
	Response Percent	Response Count
(a) supervisors	92.3%	12
(b) tasks	7.7%	1

Table 2. Judgment count for an anaphora ambiguity instance.

Ambiguity threshold. Given an anaphor, the interpretation *certainty* of a particular NP candidate is calculated as the percentage of the judgments for this NP against the total judgments for the instance. For example, consider the example in Table 2. The certainty of the NP '*supervisors*'

[6] Implemented by the NLP tool - Java WordNet Similarity Library. http://nlp.shef.ac.uk/result/software.html

[7] http://research.it.uts.edu.au/re/

is 12/13=92.3% and the certainty of the NP '*tasks*' is 1/13=7.7%. Thus, at an ambiguity threshold of, for instance, $\tau = 0.8$, the ambiguity in Table 2 is *innocuous* because the agreement between the judges exceeds the threshold.

Figure 2 shows the relationship between ambiguity threshold and occurrence of nocuous ambiguity. As in Figure 1, the number of nocuous ambiguities increases with threshold τ. For high thresholds (e.g., $\tau \geq 0.9$), more than 60% of instances are classified as nocuous. Below threshold ($\tau \leq 0.4$), fewer than 8 cases are judged nocuous. Also, comparing Figures 1 and 2 would appear to suggest that, in technical documents, anaphora ambiguity is less likely to lead to misunderstandings than coordination.

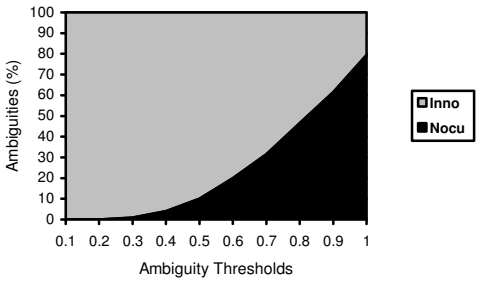


Figure 2. Proportions of interpretations at different ambiguity thresholds in the anaphora instances.

4.2 Antecedent Preference Heuristics

Drawing on the literature on anaphoric reference, we developed 12 heuristics of three types: related to *linguistic properties* of text components, to *context and discourse* information, or to *statistical* information drawn from standard corpora. Yang et al. (2010b) gives more detail. A heuristic marks candidate antecedents which it favours, or disfavours. For instance, heuristics favour definite NPs as antecedents, candidate NPs which agree in number and syntactic role with the anaphor, and those which share a syntactic collocation pattern in the text. They also favour those which respect the semantic constraints (e.g., animacy) propagated from subcategorisation information, and reward proximity to the anaphor. They disfavour candidate antecedents that occur in prepositional phrases, and those occupying a syntactic role distinct from the anaphor. Note: not all NPs are marked by all heuristics, and some heuristics are interdependent.

4.3 Nocuous Ambiguity Identification

Unlike coordination ambiguity, where judges chose for high or low modifier attachment, anaphora have scope over a variable set of potential antecedents, depending on each particular instance. To accommodate this, we developed an *antecedent* classifier which assigns a weighted antecedent tag to each NP candidate associated with an instance. Tag information is used subsequently to predict the whether the instance displays nocuous ambiguity.

The antecedent classifier is built using the Naive Bayes algorithm within the WEKA package and is trained to return three classes of candidate antecedent: *positive (Y)*, *questionable (Q)*, or *negative (N)*. In an *innocuous* case, a candidate NP will be classed as *Y* if its interpretation certainty exceeds the threshold set by τ, and tagged as *N* otherwise; in a *nocuous* case, it will be classed as *N* if its certainty is 0%, and classified as *Q* otherwise.

1. The LPS operational scenarios represent sequences of activities performed by operations personnel as **they** relate to the LPS software.		
	Response	Label
(a) the LPS operational scenarios	33.3%	Q
(b) sequences of activities	66.7%	Q
(c) activities	0%	N
(d) operations personnel	0%	N

Table 3. The determination of antecedent label for the NP candidates in a NOCUOUS ambiguity case ($\tau = 0.8$)

2. Testing performed to demonstrate to the acquirer that a CSCI system meets **its** specified requirements.		
	Response Percent	Class Label
(a) Testing	0%	N
(b) the acquirer	16.7%	N
(c) a CSCI system	83.3%	Y

Table 4. The determination of antecedent label for the NP candidates in a INNOCUOUS ambiguity case ($\tau = 0.8$)

	Antecedent Class Label		
	Y	Q	N
$\tau = 0.5$	181	54	623
$\tau = 0.6$	160	99	599
$\tau = 0.7$	137	149	572
$\tau = 0.8$	107	209	542
$\tau = 0.9$	77	261	520
$\tau = 1.0$	41	314	503

Table 5. The distribution of three antecedent class label at different ambiguity thresholds

Table 3 and 4 illustrate antecedent labels for NP antecedent candidates in a nocuous and innocuous case. Candidates (a) and (b) in Table 3 are labeled Q because their certainty falls below the threshold ($\tau = 0.8$). For the same threshold, candidate (c) in Table 4 is tagged as *Y*. Table 5

shows the distribution of tags at certainty thresholds $\tau \geq 0.5$ for all (858) candidate antecedents in our sample.

Our intended application is a system to alert experts to risk of misunderstandings. This suggests we should emphasise recall even at the expense of some precision (Berry et al. 2003). We developed two versions of the algorithm that determines whether an instance is nocuous or not, depending on the contribution made by its antecedent candidates tagged *Y*. We relax constraints by introducing two concepts: a weak positive threshold W_Y and a weak negative threshold W_N set at 0.5 and 0.4, respectively[8]. The rationale for weak thresholds is that antecedent preference reflects a spectrum with *Y* (high), *Q* (medium), and *N* (low). Weak positive and negative thresholds act as buffers to the *Q* area. Antecedent NPs that fall in the W_Y or W_N buffer area are treated as possible false negative (FN) for the classification of the label *Q*. An antecedent tag *Y/N* is labeled as weak positive or negative depending on these thresholds. The algorithm for identifying nocuous ambiguity is given in Figure 3. It treats as innocuous those cases where the antecedent label list contains one clear *Y* candidate, whose certainty exceeds all others by a margin.

Given an anaphora ambiguity instance with multiple potential NPs, the antecedent classifier returns a label list, $R = \{r_1, r_2, \ldots, r_n\}$, for individual NPs.

Parameters:
1) W_Y - the threshold for the *weak positive* label. The label *Y* is viewed as *weak positive* when the positive prediction score $r_i < W_Y$
2) W_N - the threshold for the *weak negative* label. The label *N* is viewed as *weak negative* when the negative prediction score $r_i < W_N$

Procedure:
if the label list *R* contains
(one *Y*, no *Q, one or more N*)
or
(no *Y*, one *Q*, one or more *N* but not *weak negative*)
or
(one *Y* but not *weak positive*, any number of *Q or N*)
then
the ambiguity is INNOCUOUS
else
the ambiguity is NOCUOUS

Figure 3. The algorithm for nocuous ambiguity identification

5 Experiments and Results

In all experiments, the performance was evaluated using 5-fold cross-validation, using standard measures of Precision (P), Recall (R), F-measure (F), and Accuracy. We use two naive baselines: BL-1 assumes that all ambiguity instances are *innocuous*; BL-2 assumes that they are all *nocuous*. For fair comparison against the baselines, for both forms of ambiguity, we only report the performance of our ML-based models when the incidence of nocuous ambiguities falls between 10% ~ 90% of the set (see Figures 1 and 2). We first report our findings for the identification of nocuous coordination ambiguities and then discuss the effectiveness of our model in distinguishing possible nocuous ambiguities from a set of ambiguity instances.

5.1 Nocuous Coordination Ambiguity Identification

Willis et al (2008) demonstrated the ability of their approach to adapt to different thresholds by plotting results against the two naïve base lines. Since we extended and refined their approach described we plot our experimental results (CM-1), for comparison, using the same measures, against their evaluation data (CM-2), in Figure 4.

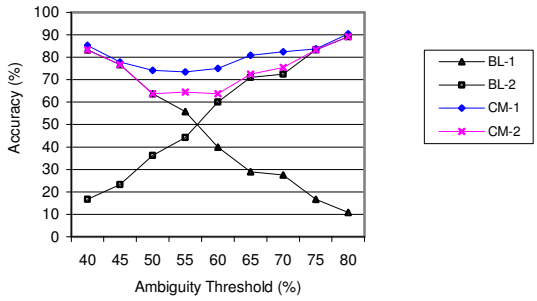


Figure 4. The performance comparison of the ML-based models, CM-1 and CM-2, to the two baseline models, BL-1 and BL-2, in nocuous coordination ambiguity identification.

Our CM-1 model performed well with an accuracy of above 75% on average at all ambiguity threshold levels. As expected, at very high and very low thresholds, we did not improve on the naive baselines (which have perfect recall and hence high accuracy). The CM-1 model displayed its advantage when the ambiguity threshold fell in the range between 0.45 and 0.75 (a significantly wider range than reported for CM-2 Willis et al (2008)). CM-1 maximum improvement was achieved around the 58% crossover point where the two naïve baselines intersect and our model achieved around 21% increased accu-

[8] Weak positive and negative thresholds are set experimentally.

racy. This suggests that the combined heuristics do have strong capability of distinguishing nocuous from innocuous ambiguity at the weakest region of the baseline models.

Figure 4 also shows that, the CM-1 model benefitted from the extended heuristics and the LogitBoost algorithm with an increased accuracy of around 5.54% on average compared with CM-2. This suggests that local context information and semantic relationships between coordinating conjuncts provide useful clues for the identification of nocuous ambiguity. Furthermore, the LogitBoost algorithm is more suitable for dealing with a numeric-attribute feature vector than the previous Logistic Regression algorithm.

5.2 Nocuous Anaphora Ambiguity Identification

We report on two implementations: one with weak thresholds (AM-1) and one without (AM-2). We compare both approaches using the baselines, BL-1 and BL-2 (in Figure 5). It shows that AM-1 and AM-2 achieve consistent improvements on baseline accuracy at high thresholds ($\tau \geq 0.75$). Here also, the improvement maximises around the 83% threshold point where the two baselines intersect. However, the ML-based models perform worse than BL-1 at the lower thresholds ($0.5 \leq \tau \leq 0.7$). One possible explanation is that, at low thresholds, performance is affected by lack of data for training of the *Q* class label, an important indicator for nocuous ambiguity (see Table 5). This is also consistent with the ML models performing well at higher thresholds, when enough nocuous instances are available for training.

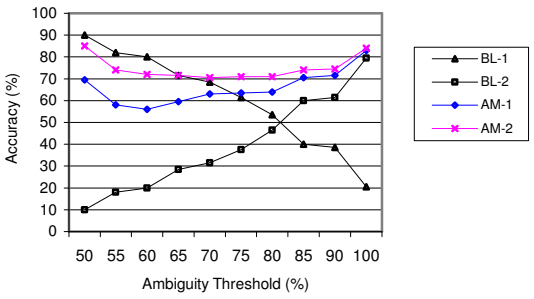


Figure 5. The performance comparison of the ML-based models, AM-1 and AM-2, to the two baseline models, BL-1 and BL-2, in nocuous anaphora ambiguity identification.

Figure 5 further shows that the model with weak thresholds (AM-1) did not perform as well as the model without weak thresholds (AM-2) on accuracy. Although both models perform much better than the baselines on precision (more experimental results are reported in Yang et al. (2010b)), the actual precisions for both models are relatively low, ranging from 0.3 ~ 0.6 at different thresholds. When the AM-1 model attempts to discover more nocuous instances using weak thresholds, it also introduces more false positives (innocuous instances incorrectly classed as nocuous). The side-effect of introducing false positives for AM-1 is to lower accuracy. However, the AM-1 model outperforms both AM-2 and BL-2 models on F-measure (Figure 6), with an average increase of 5.2 and 3.4 percentage points respectively. This reveals that relaxing sensitivity to the ambiguity threshold helps catch more instances of nocuous anaphora ambiguity.

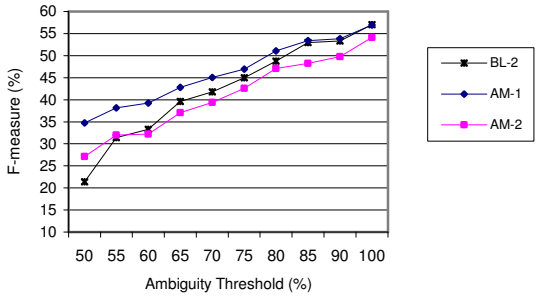


Figure 6. The performance comparison of the ML-based models, AM-1 and AM-2, to the baseline model BL-2 (naïve nocuous)

6 Discussions

We presented judges with sentences containing ambiguities without any surrounding context, even though contextual information (e.g., discourse focus) clearly contributes to interpretation. This is a weakness in our data collection technique. Besides contextual information, van Deemter's Principle of Idiosyncratic Interpretation (1998) suggests that some factors, including the reader's degree of language competence, can affect perceptions of ambiguity. Similarly, familiarity with a domain, including tacit specialist information (Polanyi, 1966), and the extent to which this is shared by a group, will have an effect on the extent to which stakeholders arrive at diverging interpretations.

In our case, we extracted instances from requirements documents covering several techni-

cal domains. Judgements are sensitive to the backgrounds of the participants, and the extent to which stakeholder groups share such a background. Also, we used several large, generic NL resources, including the BNC and WordNet. The performance of several heuristics would change if they drew on domain specific resources. Different interpretations may be compatible, and so not necessarily contribute to misunderstanding.

Finally, we used different machine learning algorithms to tackle different types of ambiguity instances: LogitBoost for coordination ambiguity and Naive Bayes for anaphora ambiguity. The main reason is that coordination heuristics returned numeric values, whereas the anaphora heuristics were Boolean. Our method assumes tailoring of the ML algorithm to the choice of heuristic. These limitations indicate that the methodology has a high degree of flexibility, but also that it has several interdependent components and background assumptions that have to be managed if an application is to be developed.

7 Related Work

Many researchers have remarked on the fact that some ambiguities are more likely than others to lead to misunderstandings, and suggested classifying them accordingly. Poesio (1996) discussed cases where multiple readings are intended to coexist, and distinguished between language inherent and human disambiguation factors from a philosophical perspective. His notion of '*perceived ambiguity*' suggests that human perceptions are what actually cause an ambiguity to be misunderstood. Van Deemter's (2004) '*vicious ambiguity*' refers to an ambiguity that has no single, strongly preferred interpretation. He proposed quantifying 'viciousness' using probabilities taken from corpus data. Van Rooy (2004) defined a notion of '*true ambiguity*': a sentence is truly ambiguous only if there are at least two interpretations that are optimally relevant. These last two approaches rely on probability analysis of language usage, and not directly on human perception, which we believe to be the key to evaluating ambiguity. Our work differs in that it takes into account the distribution of interpretations arrived at by a group of human judges engaged with a text. Our model treats ambiguity not as a property of a linguistic construct or a text, or a relation between a text and the perceptions of a single reader, but seeks to understand the mechanisms that lead to misunderstandings between people in a group or process.

Poesio *et al* (2006) have pointed out that disambiguation is not always necessary; for instance, in some complex anaphora cases, the final interpretation may not be fully specified, but only 'good enough'. Our work does not attempt disambiguation. It seeks to highlight the risk of multiple interpretations (whatever those are).

8 Conclusions and Future Work

We have presented a general methodology for automatically identifying nocuous ambiguity (i.e. cases of ambiguity where there is a risk that people will hold different interpretations) relative to some tolerance level set for such a risk. The methodology has been implemented in a ML based architecture, which combines a number of heuristics each highlighting factors which may affect how humans interpret ambiguous constructs. We have validated the methodology by identifying instances of nocuous ambiguity in coordination and anaphoric constructs. Human judgments were collected in a dataset used for training the ML algorithm and evaluation. Results are encouraging, showing an improvement of approximately 21% on accuracy for coordination ambiguity and about 3.4% on F-measure for anaphora ambiguity compared with naive baselines at different ambiguity threshold levels. We showed, by comparison with results reported in Willis et al (2008) that the methodology can be fine tuned, and extended to other ambiguity types, by including different heuristics.

Our method can highlight the risk of different interpretations arising: this is not a task a single human could perform, as readers typically have access only to their own interpretation and are not routinely aware that others hold a different one. Nonetheless, our approach has limitations, particularly around data collection, and for anaphora ambiguity at low thresholds. We envisage further work on the implementation of ambiguity tolerance thresholds

Several interesting issues remain to be investigated to improve our system's performance and validate its use in practice. We need to explore how to include different and complex ambiguity types (e.g., PP attachment and quantifier scop-

ing), and investigate whether these are equally amenable to a heuristics based approach.

Acknowledgement

This work is supported financially by UK EPSRC for the MaTREx project (EP/F068859/1), and Irish SFI for the grant 03/CE2/I303_1.

References

Daniel M. Berry, Erik Kamsties, and Michael M. Krieger. 2003. From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity. Technical Report, School of Computer Science, University of Waterloo.

Stephen Boyd, Didar Zowghi, and Alia Farroukh. 2005. Measuring the Expressiveness of a Constrained Natural Language: An Empirical Study. In *Proceedings of the 13th IEEE International Conference on Requirements Engineering (RE'05)*, Washington, DC, pages 339-52.

Eric Brill and Philip Resnik. 1994. A Rule-Based Approach to Prepositional Phrase Attachment Disambiguation. In *Proceedings of the 15th International Conference on Computational Linguistics*, pages 1198-204.

Francis Chantree, Bashar Nuseibeh, Anne de Roeck, and Alistair Willis. 2006. Identifying Nocuous Ambiguities in Natural Language Requirements. In *Proceedings of 14th IEEE International Requirements Engineering Conference (RE'06)*, Minneapolis, USA, pages 59-68.

Adam Kilgarriff. 2003. Thesauruses for Natural Language Processing. In *Proceedings of NLP-KE*, pages 5-13.

Preslav Nakov and Marti Hearst. 2005. Using the Web as an Implicit Training Set: Application to Structural Ambiguity Resolution. In *Proceedings of HLT-NAACL'05*, pages 835-42.

Akitoshi Okumura and Kazunori Muraki. 1994. Symmetric Pattern Matching Analysis for English Coordinate Structures. In *Proceedings of the 4th Conference on Applied Natural Language Processing*, pages 41-46.

Massimo Poesio. 1996. Semantic Ambiguity and Perceived Ambiguity In *Semantic Ambiguity and Underspecification* edited by K. van Deemter and S. Peters, pages 159-201.

Massimo Poesio and Ron Artstein. 2008. Introduction to the Special Issue on Ambiguity and Semantic Judgements. *Research on Language & Computation* 6: 241-45.

Massimo Poesio, Patick Sturt, Ron Artstein, and Ruth Filik. 2006. Underspecification and Anaphora: Theoretical Issues and Preliminary Evidence. *Discourse Processes* 42(2): 157-75.

Michael Polanyi. 1966. *The Tacit Dimension*. RKP, London.

Kees van Deemter. 1998. Ambiguity and Idiosyncratic Interpretation. *Journal of Semantics* 15(1): 5-36.

Kees van Deemter. 2004. Towards a Probabilistic Version of Bidirectional Ot Syntax and Semantics. *Journal of Semantics* 21(3): 251-80.

Robert van Rooy. 2004. Relevance and Bidirectional Ot. In *Optimality Theory and Pragmatic*, edited by R. Blutner and H. Zeevat, pages 173-210.

Thomas Wasow, Amy Perfors, and David Beaver. 2003. The Puzzle of Ambiguity. In *Morphology and the Web of Grammar: Essays in Menory of Steven G. Lapointe*, edited by O. Orgun and P. Sells.

Alistair Willis, Francis Chantree, and Anne De Roeck. 2008. Automatic Identification of Nocuous Ambiguity. *Research on Language & Computation* 6(3-4): 1-23.

Hui Yang, Alistair Willis, Anne de Roeck, and Bashar Nuseibeh. 2010a. Automatic Detection of Nocuous Coordination Ambiguities in Natural Language Requirements. *In Proceedings of the 25th IEEE/ACM International Conference on Automated Software Engineering Conference (ASE'10)*. (In press)

Hui Yang, Anne de Roeck, Alistair Willis, and Bashar Nuseibeh. 2010b. Extending Nocuous Ambiguity Analysis for Anaphora in Natural Language Requirements. *In Proceedings of the 18th International Requirements Engineering Conference (RE'10)*. (In press)

Contextual Modeling for Meeting Translation Using Unsupervised Word Sense Disambiguation

Yang Mei
Department of Electrical Engineering
University of Washington
yangmei@u.washington.edu

Katrin Kirchhoff
Department of Electrical Engineering
University of Washington
katrin@ee.washington.edu

Abstract

In this paper we investigate the challenges of applying statistical machine translation to meeting conversations, with a particular view towards analyzing the importance of modeling contextual factors such as the larger discourse context and topic/domain information on translation performance. We describe the collection of a small corpus of parallel meeting data, the development of a statistical machine translation system in the absence of genre-matched training data, and we present a quantitative analysis of translation errors resulting from the lack of contextual modeling inherent in standard statistical machine translation systems. Finally, we demonstrate how the largest source of translation errors (lack of topic/domain knowledge) can be addressed by applying document-level, *unsupervised* word sense disambiguation, resulting in performance improvements over the baseline system.

1 Introduction

Although statistical machine translation (SMT) has made great progress over the last decade, most SMT research has focused on the translation of structured input data, such as newswire text or parliamentary proceedings. Spoken language translation has mostly concentrated on two-person dialogues, such as travel expressions or patient-provider interactions in the medical domain. Recently, more advanced spoken-language data has been addressed, such as speeches (Stüker et al., 2007), lectures (Waibel and Fügen, 2008), and broadcast conversations (Zheng et al., 2008). Problems for machine translation in these genres include the nature of spontaneous speech input (e.g. disfluencies, incomplete sentences, etc.) and the lack of high-quality training data. Data that match the desired type of spoken-language interaction in topic, domain, and, most importantly, in style, can only be obtained by transcribing and translating conversations, which is a costly and time-consuming process. Finally, many spoken-language interactions, especially those involving more than two speakers, rely heavily on the participants' shared contextual knowledge about the domain and topic of the discourse, relationships between speakers, objects in the real-world environment, past interactions, etc. These are typically not modelled in standard SMT systems.

The problem of speech disfluencies has been addressed by disfluency removal techniques that are applied prior to translation (Rao et al., 2007; Wang et al., 2010). Training data sparsity has been addressed by adding data from out-of-domain resources (e.g. (Matusov et al., 2004; Hildebrandt et al., 2005; Wu et al., 2008)), exploiting comparable rather than parallel corpora (Munteanu and Marcu, 2005), or paraphrasing techniques (Callison-Burch et al., 2006). The lack of contextual modeling, by contrast, has so far not been investigated in depth, although it is a generally recognized problem in machine translation. Early attempts at modeling contextual information in machine translation include (Mima et al., 1998), where information about the role, rank and gender of speakers and listeners was utilized in a transfer-based spoken-language translation system for travel dialogs. In (Kumar et al., 2008)

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1227–1235,
Beijing, August 2010

statistically predicted dialog acts were used in a phrase-based SMT system for three different dialog tasks and were shown to improve performance. Recently, contextual source-language features have been incorporated into translation models to predict translation phrases for traveling domain tasks (Stroppa et al., 2007; Haque et al., 2009). However, we are not aware of any work addressing contextual modeling for statistical translation of spoken meeting-style interactions, not least due to the lack of a relevant corpus.

The first goal of this study is to provide a quantitative analysis of the impact of the lack of contextual modeling on translation performance. To this end we have collected a small corpus of parallel multi-party meeting data. A baseline SMT system was trained for this corpus from freely available data resources, and contextual translation errors were manually analyzed with respect to the type of knowledge sources required to resolve them. Our analysis shows that the largest error category consists of word sense disambiguation errors resulting from a lack of topic/domain modeling. In the second part of this study we therefore present a statistical way of incorporating such knowledge by using a graph-based unsupervised word sense disambiguation algorithm at a *global* (i.e. document) level. Our evaluation on real-world meeting data shows that this technique improves the translation performance slightly but consistently with respect to position-independent word error rate (PER).

2 Data

2.1 Parallel Conversational Data

For our investigations we used a subset of the AMI corpus (McCowan, 2005), which is a collection of multi-party meetings consisting of approximately 100 hours of multimodal data (audio and video recordings, slide images, data captured from digital whiteboards, etc.) with a variety of existing annotations (audio transcriptions, topic segmentations, summaries, etc.). Meetings were recorded in English and fall into two broad types: scenario meetings, where participants were asked to act out roles in a pre-defined scenario, and non-scenario meetings where participants were not restricted by role assignments. In the first case, the scenario was a project meeting about the development of a new TV remote control; participant roles were project manager, industrial designer, marketing expert, etc. The non-scenario meetings are about the move of an academic lab to a new location on campus. The number of participants is four. For our study we selected 10 meetings (5 scenario meetings and 5 non-scenario meetings) and had their audio transcriptions translated into German (our chosen target language) by two native speakers each. Translators were able to simultaneously read the audio transcription of the meeting, view the video, and listen to the audio, when creating the translation. The translation guidelines were designed to obtain translations that match the source text as closely as possible in terms of style – for example, translators were asked to maintain the same level of colloquial as opposed to formal language, and to generally ensure that the translation was pragmatically adequate. Obvious errors in the source text (e.g. errors made by non-native English speakers among the meeting participants) were not rendered by equivalent errors in the German translation but were corrected prior to translation. The final translations were reviewed for accuracy and the data were filtered semi-automatically by eliminating incomplete sentences, false starts, fillers, repetitions, etc. Although these would certainly pose problems in a real-world application of spoken language translation, the goal of this study is not to analyze the impact of speech-specific phenomena on translation performance (which, as discussed in Section 1, has been addressed before) but to assess the impact of contextual information such as discourse and knowledge of the real-world surroundings. Finally, single-word utterances such as *yeah, oh, no, sure*, etc. were downsampled since they are trivial to translate and were very frequent in the corpus; their inclusion would therefore bias the development and tuning of the MT system towards these short utterances at the expense of longer, more informative utterances.

Table 1 shows the word counts of the translated meetings after the preprocessing steps described above. As an indicator of inter-translator

ID	type	# utter.	# word	S-BLEU
ES2008a	S	224	2327	21.5
IB4001	NS	419	3879	24.5
IB4002	NS	447	3246	30.5
IB4003	NS	476	5118	24.1
IB4004	NS	593	5696	26.9
IB4005	NS	381	4719	30.4
IS1008a	S	191	2058	25.8
IS1008b	S	353	3661	24.1
IS1008c	S	308	3351	19.6
TS3005a	S	245	2339	28.1

Table 1: Sizes and symmetric BLEU scores for translated meetings from the AMI corpus (S = scenario meeting, NS = non-scenario meeting).

agreement we computed the symmetric BLEU (S-BLEU) scores on the reference translations (i.e. using one translation as the reference and the other as the hypothesis, then switching them and averaging the results). As we can see, scores are fairly low overall, indicating large variation in the translations. This is due to (a) the nature of conversational speech, and (b) the linguistic properties of the target language. Conversational data contain a fair amount of colloquialisms, referential expressions, etc. that can be translated in a variety of ways. Additionally, German as the target language permits many variations in word order that convey slight differences in emphasis, which is turn is dependent on the translators' interpretation of the source sentence. German also has rich inflectional morphology that varies along with the choice of words and word order (e.g. verbal morphology depends on which subject is chosen).

2.2 SMT System Training Data

Since transcription and translation of multi-party spoken conversations is extremely time-consuming and costly, it is unlikely that parallel conversational data will ever be produced on a sufficiently large scale for a variety of different meeting types, topics, and target languages. In order to mimic this situation we trained an initial English-German SMT system on freely available out-of-domain data resources. We considered the following parallel corpora: news text (de-news[1], 1.5M words), EU parliamentary proceedings (Europarl (Koehn, 2005), 24M words) and EU legal documents (JRC Acquis[2], 35M words), as well as two generic English-German machine-readable dictionaries[3],[4] (672k and 140k entries, respectively).

3 Translation Systems

We trained a standard statistical phrase-based English-German translation system from the resources described above using Moses (Hoang and Koehn, 2008). Individual language models were trained for each data source and were then linearly interpolated with weights optimized on the development set. Similarly, individual phrase tables were trained and were then combined into a single table. Binary indicator features were added for each phrase pair, indicating which data source it was extracted from. Duplicated phrase pairs were merged into a single entry by averaging their scores (geometric mean) over all duplicated entries. The weights for binary indicator features were optimized along with all other standard features on the development set. Our previous experience showed that this method worked better than the two built-in features in Moses for handling multiple translation tables. We found that the JRC corpus obtained very small weights; it was therefore omitted from further system development. Table 2 reports results from six different systems: the first (System 1) is a system that only uses the parallel corpora but not the external dictionaries listed in Section 2.2. System 2 additionally uses the external dictionaries. All systems use two meetings (IB4002 and IS1008b) as a development set for tuning model parameters and five meetings for testing (IB4003-5,IS1008c,TS3005a). For comparison we also trained a version of the system where a small in-domain data set (meetings ES2008a, IB4001, and IS1008a) was added to the training data (System 3). Finally, we also compared our performance against Google Translate, which is a state-of-the-art statistical MT system with unconstrained ac-

[1] www.iccs.inf.ed.ac.uk/~pkoehn/publications/de-news
[2] http://wt.jrc.it/lt/Acquis/
[3] http://www.dict.cc
[4] http://www-user.tu-chemnitz.de/~fri/ding

	System description	Dev set				Eval set			
		OOV (%)		Trans. Scores		OOV (%)		Trans. Scores	
		EN	DE	BLEU	PER	EN	DE	BLEU	PER
System 1	OOD parallel data only	4.1	17.0	23.8	49.0	6.5	20.5	21.1	49.5
System 2	System 1 + dictionaries	1.5	15.9	24.6	47.3	2.8	16.3	21.7	48.4
System 3	System 1 + ID parallel data	3.5	13.4	24.7	47.2	5.8	19.7	21.9	48.3
System 4	System 2 + ID parallel data	1.2	12.9	25.4	46.1	2.5	15.9	22.0	48.2
System 5	System 4 + web data	1.2	12.8	26.0	45.9	2.5	15.8	22.1	48.1
System 6	Google Translate	–	–	25.1	49.1	–	–	23.7	50.8

Table 2: System performance using out-of-domain (OOD) parallel data only vs. combination with a small amount of in-domain (ID) data and generic dictionaries. For each of the development (DEV) and evaluation (Eval) set, the table displays the percentages of unknown word types (OOV) for English (EN) and German (DE), as well as the translation scores of BLEU (%) and PER.

cess to the web as training data (System 6). As expected, translation performance is fairly poor compared to the performance generally obtained on more structured genres. The use of external dictionaries helps primarily in reducing PER scores while BLEU scores are only improved noticeably by adding in-domain data. System 6 shows a more even performance across dev and eval sets than our trained system, which may reflect some degree of overtuning of our systems to the relatively small development set (about 7K words). However, the PER scores of System 6 are significantly worse compared to our in-house systems.

In order to assess the impact of adding web data specifically collected to match our meeting corpus we queried a web portal[5] that searches a range of English-German bilingual web resources and returns parallel text in response to queries in either English or German. As queries we used English phrases from our development and evaluation sets that (a) did not already have phrasal translations in our phrase tables, (b) had a minimum length of four words, and (c) occurred at least twice in the test data. In those cases where the search engine returned results with an exact match on the English side, we word-aligned the resulting parallel text (about 600k words) by training the word alignment together with the news text corpus. We then extracted new phrase pairs (about 3k) from the aligned data. The phrasal scores assigned to the new phrase pairs were set to 1; the lexical scores were computed from a word lexicon trained over both the baseline data resources and the parallel web data. However, results (Row 5 in Table 2) show that performance hardly improved, indicating the difficulty in finding matching data sources for conversational speech.

Table 2 also shows the impact of different data resources on the percentages of unknown word types (OOV) for both the source and target languages. The use of external dictionaries gave the largest reduction of OOV rates (System 1 vs. System 2 and System 3 vs. System 4), followed by the use of in-domain data (System 1 vs. System 3 and System 2 vs. System 4). Since they were retrieved by multi-word query phrases, adding the web data did not lead to significant reduction on the OOV rates (System 4 vs. System 5).

Finally, we also explored a hierarchical phrase-based system as an alternative baseline system. The system was trained using the Joshua toolkit (Li et al., 2009) with the same word alignments and language models as were used in the standard phrase-based baseline system (System 4). After extracting the phrasal (rule) tables for each data source, they were combined into a single phrasal (rule) table using the same combination approach as for the basic phrase-based system. However, the translation results (BLEU/PER of 24.0/46.6 (dev) and 20.8/47.6 (eval), respectively) did not show any improvement over the basic phrase-based system.

[5]http://www.linguee.com

4 Analysis of Baseline Translations: Effect of Contextual Information

The output from System 5 was analyzed manually in order to assess the importance of modeling contextual information. Our goal was not to determine how translation of meeting style data can be improved in general – better translations could certainly be generated by better syntactic modeling, addressing morphological variation in German, and generally improving phrasal coverage, in particular for sentences involving colloquial expressions. However, these are fairly general problems of SMT that have been studied previously. Instead, our goal was to determine the relative importance of modeling different contextual factors, such as discourse-level information or knowledge of the real-world environment, which have not been studied extensively.

We considered three types of contextual information: discourse coherence information (in particular anaphoric relations), knowledge of the topic or domain, and real-world/multimodal information. Anaphoric relations affect the translation of referring expressions in cases where the source and target languages make different grammatical distinctions. For example, German makes more morphological distinctions in noun phrases than English. In order to correctly translate an expression like "the red one" the grammatical features of the target language expression for the referent need to be known. This is only possible if a sufficiently large context is taken into account during translation and if the reference is resolved correctly. Knowledge of the topic or domain is relevant for correctly translating content words and is closely related to the problem of word sense disambiguation. In our current setup, topic/domain knowledge could be particularly helpful because in-domain training data is lacking and many word translations are obtained from generic dictionaries that do not assign probabilities to competing translations. Finally, knowledge of the real-world environment, such as objects in the room, other speakers present, etc. determines translation choices. If a speaker utters the expression "that one" while pointing to an object, the correct translation might depend on the grammatical features of the linguistic expression for that object; e.g. in German, the translation could be "die da", "der da" or "das da". Since the participants in our meeting corpus use slides and supporting documents we expect to see some effect of such exophoric references to external objects.

Error type	% (dev)	% (eval)
Word sense	64.5	68.2
Exophora (addressee)	24.3	23.4
Anaphora	10.2	7.8
Exophora (other)	1.0	0.6

Table 3: Relative frequency of different error types involving contextual knowledge. The total number of errors is 715, for 315 sentences.

In order to quantify the influence of contextual information we manually analyzed the 1-best output of System 5, identified those translation errors that require knowledge of the topic/domain, larger discourse, or external environment for their resolution, classified them into different categories, and computed their relative frequencies. We then corrected these errors in the translation output to match at least one of the human references, in order to assess the maximum possible improvement in standard performance scores that could be obtained from contextual modeling. The results are shown in Tables 3 and 4. We observe that out of all errors that can be related to the lack of contextual knowledge, word sense confusions are by far the most frequent. A smaller percentage of errors is caused by anaphoric expressions. Contrary to our expectations, we did not find a strong impact of exophoric references; however, there is one crucial exception where real-world knowledge does play an important role. This is the correct translation of the addressee *you*. In English, this form is used for the second person singular, second person plural, and the generic interpretation (as in "one", or "people"). German has three distinct forms for these cases and, additionally, formal and informal versions of the second-person pronouns. The required formal/informal pronouns can only be determined by prior knowledge of the relationships among the meeting participants. However, the singular-plural-generic distinction can potentially be resolved by multimodal informa-

	Original		Corrected	
	BLEU (%)	PER	BLEU (%)	PER
dev	26.0	45.9	27.5	44.0
eval	22.1	48.1	23.3	46.0

Table 4: Scores obtained by correcting errors due to lack of contextual knowledge.

tion such as gaze, head turns, body movements, or hand gestures of the current speaker. Since these errors affect mostly single words as opposed to larger phrases, the impact of the corrections on BLEU/PER scores is not large. However, for practical applications (e.g. information extraction or human browsing of meeting translations) the correct translation of content words and referring expressions would be very important. In the remainder of the paper we therefore describe initial experiments designed to address the most important source of contextual errors, viz. word sense confusions.

5 Resolving Word Sense Disambiguation Errors

The problem of word sense disambiguation (WSD) in MT has received a fair amount of attention before. Initial experiments designed at integrating a WSD component into an MT system (Carpuat and Wu, 2005) did not meet with success; however, WSD was subsequently demonstrated to be successful in data-matched conditions (Carpuat and Wu, 2007; Chan et al., 2007). The approach pursued by these latter approaches is to train a supervised word sense classifier on different phrase translation options provided by the phrase table of an initial baseline system (i.e. the task is to separate different phrase senses rather than word senses). The input features to the classifier consist of word features obtained from the immediate context of the phrase in questions, i.e. from the same sentence or from the two or three preceding sentences. The classifier is usually trained only for those phrases that are sufficiently frequent in the training data.

By contrast, our problem is quite different. First, many of the translation errors caused by choosing the wrong word sense relate to words obtained from an external dictionary that do not occur in the parallel training data; there is also little in-domain training data available in general. For these reasons, training a supervised WSD module is not an option without collecting additional data. Second, the relevant information for resolving a word sense distinction is often not located in the immediately surrounding context but it is either at a more distant location in the discourse, or it is part of the participants' background knowledge. For example, in many meetings the opening remarks refer to slides and an overhead projector. It is likely that subsequent mentioning of *slide* later on during the conversation also refer to overhead slides (rather than e.g. *slide* in the sense of "playground equipment"), though the contextual features that could be used to identify this word sense are not located in the immediately preceding sentences. Thus, in contrast to supervised, local phrase sense disambiguation employed in previous work, we propose to utilize unsupervised, *global* word sense disambiguation, in order to obtain better modeling of the topic and domain knowledge that is implicitly present in meeting conversations.

5.1 Unsupervised Word Sense Disambiguation

Unsupervised WSD algorithms have been proposed previously (e.g. (Navigli and Lapata, 2007; Cheng et al., 2009)). The general idea is to exploit measures of word similarity or relatedness to jointly tag all words in a text with their correct sense. We adopted the graph-based WSD method proposed in (Sinha and Mihalcea, 2007), which represents all word senses in a text as nodes in an undirected graph $G = (V, E)$. Pairs of nodes are linked by edges weighted by scores indicating the similarity or relatedness of the words associated with the nodes. Given such a graph, the likelihood of each node is derived by the PageRank algorithm (Brin and Page, 1998), which measures the relative importance of each node to the entire graph by considering the amount of "votes" the node receives from its neighboring nodes. The PageRank algorithm was originally designed for directed graphs, but can be easily extended to an undirected graph. Let $PR(v_i)$ denote the PageRank score of v_i. The PageRank algorithm itera-

tively updates this score as follows:

$$PR(v_i) = (1-d) + d \sum_{(v_i,v_j)\in E} PR(v_j) \frac{w_{ij}}{\sum_k w_{kj}}$$

where w_{ij} is the similarity weight of the undirected edge (v_i, v_j) and d is a damping factor, which is typically set to 0.85 (Brin and Page, 1998). The outcome of the PageRank algorithm is numerical weighting of each node in the graph. The sense with the highest score for each word identifies its most likely word sense. For our purposes, we modified the procedure as follows. Given a document (meeting transcription), we first identify all content words in the source document. The graph is then built over all target-language translation candidates, i.e. each node represents a word translation. Edges are then established between all pairs of nodes for which a word similarity measure can be obtained.

5.2 Word Similarity Measures

We follow (Zesch et al., 2008a) in computing the semantic similarity of German words by exploiting the Wikipedia and Wiktionary databases. We use the publicly available toolkits JWPL and JWKTL (Zesch et al., 2008b) to retrieve relevant articles in Wikipedia and entries in Wiktionary for each German word – these include the first paragraphs of Wikipedia articles entitled by the German word, the content of Wiktionary entries of the word itself as well as of closely related words (hypernyms, hyponyms, synonyms, etc.). We then concatenate all retrieved material for each word to construct a pseudo-gloss. We then lowercase and lemmatize the pseudo-glosses (using the lemmatizer available in the TextGrid package [6]), exclude function words by applying a simple stop-word list, and compute a word similarity measure for a given pair of words by counting the number of common words in their glosses.

We need to point out that one drawback in this approach is the low coverage of German content words in the Wikipedia and Wiktionary databases. Although the English edition contains millions of entries, the German edition of Wikipedia and Wiktionary is much smaller – the coverage of all content words in our task ranges between 53% and 56%, depending on the meeting, which leads to graphs with roughly 3K to 5K nodes and 8M to 13M edges. Words that are not covered mostly include rare words, technical terms, and compound words.

5.3 Experiments and Results

For each meeting, the derived PageRank scores were converted into a positive valued feature, referred to as the WSD feature, by normalization and exponentiation:

$$f_{WSD}(w_g|w_e) = \exp\left\{\frac{PR(w_g)}{\sum_{w_g\in H(w_e)} PR(w_g)}\right\}$$

where $PR(w_g)$ is the PageRank score for the German word w_g and $H(w_e)$ is the set of all translation candidates for the English word w_e. Since they are not modeled in the graph-based method, multi-words phrases and words that are not found in the Wikipedia or Wiktionary databases will receive the default value 1 for their WSD feature. The WSD feature was then integrated into the phrase table to perform translation. The new system was optimized as before.

It should be emphasized that the standard measures of BLEU and PER give an inadequate impression of translation quality, in particular because of the large variation among the reference translations, as discussed in Section 4. In many cases, better word sense disambiguation does not result in better BLEU scores (since higher gram matches are not affected) or even PER scores because although a feasible translation has been found it does not match any words in the reference translations. The best way of evaluating the effect of WSD is to obtain human judgments – however, since translation hypotheses change with every change to the system, our original error annotation described in Section 4 cannot be re-used, and time and resource constraints prevented us from using manual evaluations at every step during system development.

In order to loosen the restrictions imposed by having only two reference translations, we utilized a German thesaurus[7] to automatically extend the content words in the references with synonyms. This can be seen as an automated way of

[6] http://www.textgrid.de/en/beta.html

[7] http://www.openthesaurus.de

	No WSD			With WSD		
	BLEU (%)	PER	XPER	BLEU (%)	PER	XPER
dev	25.4	46.1	43.4	25.4	45.6	42.9
eval	22.0	48.2	44.6	22.0	47.9	44.0
IB4003	21.4	48.3	44.4	21.4	47.5	43.8
IB4004	22.4	48.5	44.4	23.1	48.4	43.9
IB4005	25.4	45.9	42.4	25.3	45.6	42.2
IS1008c	15.9	52.9	50.0	14.9	52.3	48.6
TS3005a	23.1	45.2	41.9	23.2	45.3	41.7

Table 5: Performance of systems with and without WSD for dev and eval sets as well as individual meetings in the eval set.

approximating the larger space of feasible translations that could be obtained by producing additional human references. Note that the thesaurus provided synonyms for only roughly 50% of all content words in the dev and eval set. For each of them, on average three synonyms are found in the thesaurus. We use these extended references to recompute the PER score as an indicator of correct word selection. All results (BLEU, PER and extended PER (or XPER)) are shown in Table 5. As expected, BLEU is not affected but WSD improves the PER and XPER slightly but consistently. Note that this is despite the fact that only roughly half of all content words received disambiguation scores.

Finally, we provide a concrete example of translation improvements, with improved words highlighted:
Source:
on the balcony
there's that **terrace**
there's no place inside the building
Translation, no WSD:
auf dem balkon
es ist das **absatz**
es gibt keinen platz innerhalb des gebäudes
Translation, with WSD:
auf dem balkon
es ist das **terrasse**
es gibt keinen platz gebäudeintern
References:
auf dem balkon / auf dem balkon
da gibt es die **terrasse** */ da ist die* **terrasse**
es gibt keinen platz im gebäude / es gibt keinen platz innen im gebäude

6 Summary and Conclusions

We have presented a study on statistical translation of meeting data that makes the following contributions: to our knowledge it presents the first quantitative analysis of contextual factors in the statistical translation of multi-party spoken meetings. This analysis showed that the largest impact could be obtained in the area of word sense disambiguation using topic and domain knowledge, followed by multimodal information to resolve addressees of *you*. Contrary to our expectations, further knowledge of the real-world environment (such as objects in the room) did not show an effect on translation performance. Second, it demonstrates the application of *unsupervised, global* WSD to SMT, whereas previous work has focused on supervised, local WSD. Third, it explores definitions derived from collaborative Wiki sources (rather than WordNet or existing dictionaries) for use in machine translation. We demonstrated small but consistent improvements even though word coverage was incomplete. Future work will be directed at improving word coverage for the WSD algorithm, investigating alternative word similarity measures, and exploring the combination of global and local WSD techniques.

Acknowledgments

This work was funded by the National Science Foundation under Grant IIS-0840461 and by a grant from the University of Washington's Provost Office. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding organizations.

References

S. Brin and L. Page. 1998. "The Anatomy of a Large-Scale Hypertextual Web Search Engine". *Proceedings of WWW7*.

C. Callison-Burch, P. Koehn and M. Osborne. 2006. "Improved Statistical Machine Translation Using Paraphrases". *Proceedings of NAACL*.

M. Carpuat and D. Wu. 2005. "Word sense disambiguation vs. statistical machine translation". *Proceedings of ACL*.

M. Carpuat and D. Wu. 2007. "Improving statistical machine translation using word sense disambiguation". *Proceedings of EMNLP-CoNLL*.

Y.S. Chan and H.T. Ng and D. Chiang 2007. "Word sense disambiguation improves statistical machine translation". *Proceedings of ACL*.

P. Chen, W. Ding, C. Bowes and D. Brown. 2009. "A fully unsupervised word sense disambiguation method using dependency knowledge". *Proceedings of NAACL*.

E. Gabrilovich and S. Markovitch. 2007 "Computing semantic relatedness using Wikipedia-based explicit semantic analysis". *Proceedings of IJCAI*.

R. Haque, S.K. Naskar, Y. Ma and A. Way. 2009. "Using supertags as source language context in SMT". *Proceedings of EAMT*.

A.S. Hildebrandt, M. Eck, S. Vogel and A. Waibel. 2005. "Adaptation of the Translation Model for Statistical Machine Translation using Information Retrieval". *Proceedings of EAMT*.

H. Hoang and P. Koehn. 2008. "Design of the Moses decoder for statistical machine translation". *Proceedings of SETQA-NLP*.

P. Koehn. 2005. "Europarl: a parallel corpus for statistical machine translation". *Proceedings of MT Summit*.

V. Kumar, R. Sridhar, S. Narayanan and S. Bangalore. 2008. "Enriching spoken language translation with dialog acts". *Proceedings of HLT*.

Z. Li et al.. 2009. "Joshua: An Open Source Toolkit for Parsing-based Machine Translation". *Proceedings of StatMT*.

E. Matusov, M. Popović, R. Zens and H. Ney. 2004. "Statistical Machine Translation of Spontaneous Speech with Scarce Resources". *Proceedings of IWSLT*.

A. McCowan. 2005. "The AMI meeting corpus",

H. Mima, O. Furuse and H. Iida. 1998. "Improving Performance of Transfer-Driven Machine Translation with Extra-Linguistic Information from Context, Situation and Environment". *Proceedings of Coling*. *Proceedings of the International Conference on Methods and Techniques in Behavioral Research*.

D.S. Munteanu and D. Marcu. 2005. "Improving machine translation performance by exploiting non-parallel corpora". *Computational Linguistics*.

R. Navigli and M. Lapata. 2007. "Graph Connectivity Measures for Unsupervised Word Sense Disambiguation", *Proceedings of IJCAI*

S. Rao and I. Lane and T. Schultz. 2007. "Improving spoken language translation by automatic disfluency removal". *Proceedings of MT Summit*. 31(4).

R. Sinha and R. Mihalcea. 2007. "Unsupervised Graph-based Word Sense Disambiguation Using Measures of Word Semantic Similarity", *Proceedings of IEEE-ICSC*

N. Stroppa, A. Bosch and A. Way. 2007. "Exploiting Source Similarity for SMT using Context-Informed Features". *Proceedings of TMI*.

S. Stüker, C. Fügen, F. Kraft and M. Wölfel. 2007. "The ISL 2007 English Speech Transcription System for European Parliament Speeches". *Proceedings of Interspeech*.

A. Waibel and C. Fügen. 2008. "Spoken Language Translation – Enabling cross-lingual human-human communication". *Proceedings of Coling)*.

W. Wang, G. Tur, J. Zheng and N.F. Ayan. 2010. "Automatic disfluency removal for improving spoken language translation". *Proceedings of ICASSP*. *IEEE Signal Processing Magazine*

H. Wu, H. Wang and C. Zong. 2008. "Domain adaptation for statistical machine translation with domain dictionary and monolingual corpora",

T. Zesch, C. Müller and Iryna Gurevych. 2008. "Extracting Lexical Semantic Knowledge from Wikipedia and Wiktionary". *Proceedings of LREC*.

T. Zesch, Christof Müler and Iryna Gurevych. 2008. "Using Wiktionary for Computing Semantic Relatedness".

J. Zheng, W. Wang and N.F. Ayan. 2008. "Development of SRI's translation systems for broadcast news and broadcast conversations". *Proceedings of Interspeech*. *Proceedings of AAAI*.

A Working Report on Statistically Modeling Dative Variation in Mandarin Chinese

Yao Yao
University of California, Berkeley
Department of Linguistics
yaoyao@berkeley.edu

Feng-hsi Liu
University of Arizona
Department of East Asian Studies
fliu@u.arizona.edu

Abstract

Dative variation is a widely observed syntactic phenomenon in world languages (e.g. *I gave John a book* and *I gave a book to John*). It has been shown that which surface form will be used in a dative sentence is not a completely random choice, rather, it is conditioned by a wide range of linguistic factors. Previous work by Bresnan and colleagues adopted a statistical modeling approach to investigate the probabilistic trends in English dative alternation. In this paper, we report a similar study on Mandarin Chinese. We further developed Bresnan et al.'s models to suit the complexity of the Chinese data. Our models effectively explain away a large proportion of the variation in the data, and unveil some interesting probabilistic features of Chinese grammar. Among other things, we show that Chinese dative variation is sensitive to heavy NP shift in both left and right directions.

1 Introduction

1.1 Overview

In traditional linguistic research, the study of syntax is most concerned with grammaticality. Sentences are either grammatical or ungrammatical, and syntactic theories are proposed to explain the structural features that cause (un)grammaticality. Meanwhile, little attention has been paid to the relative acceptability of grammatical sentences. If two sentences are both grammatical and basically express the same meaning, are they equally likely to occur in the language? The answer is probably *no*. For example, in English, the sentence *I have read that book* is much more frequent than *That book I have read*. The latter topicalized sentence is only used when the entity denoted by *That book* is in focus. This indicates that the choice of surface sentence form is not entirely random, but conditioned by some factors including information status.

Thus, instead of categorizing sentences as grammatical or ungrammatical, a better way to express the degree of grammaticality would be to use a likelihood continuum, from 0 to 1, where ungrammatical sentences have zero likelihood and grammatical sentences fall somewhere between 0 and 1, with some being more likely than others. The idea of associating linguistic forms with various probabilities has been around for a while (see Jurafsky, 2003 and Manning, 2003 for an extensive review). Recent psycholinguistic research has shown that just like grammaticality, the likelihoods of sentence forms are also part of the user's linguistic knowledge. Sentences with high probabilities are in general easier to comprehend and produce, and their production is more prone to phonetic reduction (Bresnan, 2007; Gahl and Garnsey, 2004; Levy, 2008; among others). The famous example of garden path sentences also exemplifies the difficulty of comprehension in low-probability sentence forms.

If we accept the premise of probabilistic syntax, then an immediate question is what determines these probabilities. In the current work, we address this question by investigating a particular type of probabilistic phenomenon, i.e. dative variation in Chinese. We show that the probabilities of

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1236–1244,
Beijing, August 2010

various surface forms of Chinese dative sentences can be well estimated by a linear combination of a set of formal and semantic features.

The remainder of this paper is organized as follows. Section 1.2 briefly reviews previous work on English dative variation. Section 1.3 introduces dative variation in Chinese. Section 2 describes the dataset and the statistical models used in the current study. Section 3 presents modeling results, followed by a discussion in Section 4. Section 5 concludes the paper with a short summary. To preview the results, we show that dative variation in Chinese is more complicated than in English, in that it features two levels of variation, which exhibit different (sometimes even opposite) probabilistic patterns.

1.2 Dative variation in English

A dative sentence is a sentence that encodes a transfer event. Typical verbs of transfer in English include *give, send, mail*, etc. A characterizing property of transfer events is that they often involve two objects. In addition to the direct object (DO), the verb also takes an indirect object (IO) which usually denotes the recipient of the transfer action. For instance, in sentence 1a, the direct object is *a book* and the indirect object is *John*.

Cross-linguistically, it has been documented that many languages in the world have multiple syntactic forms for encoding the same transfer event (Margetts and Austin, 2007 , among others). In English, both 1a and 1b describe the same event, but 1a is a double object form (V IO DO) while 1b takes a prepositional phrase (V DO *to* IO).

(1) a. I gave John a book. → V IO DO
b. I gave a book to John. → V DO *to* IO

A number of conditioning factors have been identified for the alternation between the two surface forms. For instance, when the indirect object is a pronoun (e.g. *him*), it is more likely to have the double object form (i.e. *I gave him a book*) than the PP form (i.e. *I gave a book to him*). On the other hand, if the indirect object is a complex NP (with relative clauses), it tends to occur at the end of the sentence. Since most of these effects are subtle and often correlated with each other (e.g. definiteness, pronominality and syntactic complexity), investigating individual factors can give convoluted and unreliable results. To avoid this problem, many recent works in the field adopted a statistical modeling approach (Bresnan et al., 2007; Wasow and Arnold, 2003, among others). Instead of investigating separate factors, statistical models are built on large-scale datasets, using all potential conditioning factors to predict the surface form. In Bresnan et al. (2007), a dozen predictors relating to the verb (type of transfer event), the two object NPs (accessibility, pronominality, definiteness, syntactic complexity, etc), and the discourse (presence of parallel structures) were used to make the prediction. Using data input from 2,360 dative sentences from the Switchboard corpus, the model correctly predicted surface form in 97% of the sentences, which was a great improvement over the baseline prediction accuracy of 79% (i.e. the percentage of correct responses if the model knows nothing but which variant is more frequently used). It also showed that dative variation in English was indeed sensitive to all the predictors in the model.

1.3 Dative variation in Chinese

Dative variation in Chinese is much more complicated than in English. In addition to the two word orders that exist in English (2a, 2b), it is also common for direct object to appear before the verb, as in a BA construction or a topicalized sentence (2c). Besides, indirect object can also precede the verb, as shown in 2d. Another dimension of variation is in the use of coverbs *gei* and *ba*, both of which can be optional (2b, 2c; see Li and Thompson, 1981 for a detailed discussion on this), or replaced by other morphemes (*zhu, yu, jiang*, etc).

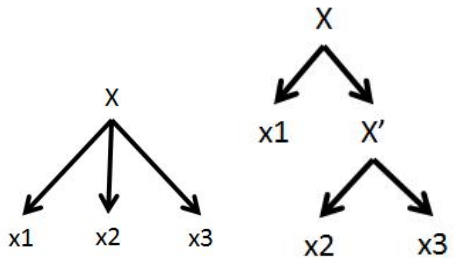
(2) a. John song-le shu gei Mary.
John give-ASP book to Mary
John gave one/some book(s) to Mary.
→ V DO IO
b. John song (gei) Mary yiben shu.
John gave (to) Mary one book
John gave Mary a book.
→ V IO DO
c. John ba shu song (gei) Mary, (ba)
John BA book gave (to) Mary (BA)

jiu song (gei) Kate.
wine gave (to) Kate
John gave the book(s) to Mary and gave the wine to Kate.
→ DO V IO

d. Ta meiren fa-le yiben shu.
He everyone allocated one book
He gave everyone a book.
→ IO V DO

For the purpose of the current study, we will ignore the existence (hence also the variation) of *gei* and *ba*, and concentrate on the variation in the relative order of V, DO and IO. In addition, our corpus search shows that sentences in the form of IO V DO are the least frequent (<9%) and mostly limited to a small set of verbs (mostly *fa* "to allocate" and *banfa* "to award"), so we drop this category from the current study. Thus the three remaining word order variants are: DO V IO, V DO IO, and V IO DO.

Generally speaking, there are two ways of modeling a variation phenomenon involving three variants. One way is to assume that the three variants are equally dissimilar from one another and the selection process is just to pick one out of three (Fig. 1a). The other approach is to assume a hierarchical structure: two of the variants are more similar to each other than they are to the third one and thus form a subcategory first before they join the third variant (Fig. 1b). In the selection process, the user first selects the subcategory (i.e. **x1** or **x'** in Fig 1b), and depending on which subcategory is chosen, they might need to make a second choice between two end nodes (i.e. **x2** and **x3**).



(a) (b)

Figure 1: Two possible schemas

We argue that the variation among the three word order variants in the current study is better modeled by a schema like Fig 1b, for both theoretical and methodological reasons. First, V DO IO and V IO DO are structurally more similar to each other than they are to DO V IO. Both V DO IO and V IO DO are in canonical word order of Chinese but the form DO V IO features the preposing (or topicalization) of the DO, whether or not the BA morpheme is present. Object preposing also exists outside ditransitive sentences (e.g. 3). Previous research has associated object preposing with the disposal meaning of the verb phrase, and the definiteness, givenness and weight of the object NP (Li and Thompson, 1981; Liu, 2007).

(3) a. Wo ba fan chi wan le.
I BA rice eat finish SEP
I have finished the rice.

b. Ta zhe dianying kan-le henduo bian.
he this movie saw many time
He has watched this movie for many times.

There is also a methodological motivation for adopting a hierarchical schema. Though it is not impossible to model a categorical variation with more than two variants (using multinomial logistic regression), binary variation is much easier to model and the interpretation of the results is more straightforward (this is especially true when random effects are present).

In view of the above, we propose the schema in Fig 2 for modeling the current variation phenomenon. We refer to sentences in the form of DO V IO as preverbal ditransitive sentences (since DO is before the verb), while both V DO IO and V IO DO are postverbal ditransitives. The distinction between the latter two forms regards whether DO is before or after IO, therefore one is termed as pre-IO and the other post-IO. Compared with the upper-level preverbal-postverbal distinction, the lower-level variation is much less studied in the literature (though see Liu, 2006 for a relevant discussion).

Corresponding to the schema in Fig 2, we constructed two separate models, one for the upper-level variation ("upper model") and the other for the lower-level variation ("lower model").

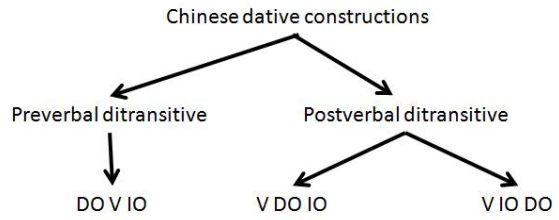


Figure 2: A two-level schema for Chinese dative variation

2 Methodology

2.1 Corpus and dataset

The data we use are from the Sinica Corpus of Modern Chinese (v3.1; Huang et al., 1995). We first compiled a list of 36 verbs that could be used ditransitively (see Appendix A) and then extracted from the corpus all sentences containing these words (n= 48,825 sentences). We then manually went through the sentences and selected those that (a) featured the ditransitive sense of the target verb, with both object NPs being overt, and (b) were in the form of any of the three form variants. 1,574 sentences remained after step (a) [1] and 1,433 after step (b) [2].

Further removal was conducted on verbs that were too sparse in the dataset. In each variation model, we removed verbs with fewer than two occurrences under either form variant. The final dataset for the upper model has 1149 sentences (of 20 verb types) while the dataset for the lower model has 801 sentences (of 14 verb types). The latter dataset is largely but not fully contained in the former due to the elimination of low-frequency verbs.

2.2 Data annotation

Similar to Bresnan et al.'s work on English, we annotated each data sentence for a wide range of features pertaining to the verb and the two NPs (see Appendix B for a complete list of annotated factors). Specifically, the verb was coded either as expressing a canonical transfer event, such as *ji* "to mail", or an extended transfer event, such as *jieshao* "to introduce". Semantic annotation of the two NPs is much trickier in Chinese than in English due to the lack of morphology. In practice, we used Bresnan et al.'s criteria for English, whenever applicable (e.g. accessibility, person, concreteness, animacy). In cases where the English rules did not apply (e.g. definiteness and number of bare NPs in Chinese), we developed working principles based on phrasal substitution. For example, if a bare NP can take a specifier like *yige/yizhi* "one" without changing sentence meaning, it is considered to be indefinite. Conversely, if a bare NP is better replaced with a full NP with a demonstrative *zhege* "this" or *nage* "that", it is coded as definite. Similar rules were used to assist annotating the number feature, using specifiers *yige/nage* "one"/"that" and *yixie/naxie* "some"/"those".

In addition to the factors in the English model, we also coded a set of structural features, including the presence of a following verb after the ditransitive construction, the presence of quantifiers/numerals in the NPs, and whether or not the ditransitive structure is embedded, nominalized, or relativized, etc. We suspect that since semantic features are often covert in Chinese words, it is possible that overt marking (e.g. the use of quantifiers/numerals) plays a more important role in conditioning surface form variation.

Finally we also included genre in the model. Sentences listed under the categories of dialogue and speech in the Sinica corpus were coded as "spoken" and the rest are coded as "written".

Altogether 24 factors were annotated and included in the statistical models as predictor variables. All variables are categorical except for the (log) length difference between DO and IO, which is numerical.

2.3 Statistical models

The statistical tool we use is mixed-effects logistic regression models. Compared with regular logistic regression models, mixed-effects models are more sophisticated in that they allow the user to specify factors that might introduce ran-

[1] A vast number of sentences were removed because the target verb was not used as a verb, or used with a different sense, or used as part of a different verb phrase, e.g. *fa* to allocate could also mean to bloom or be used in *fazhan* to develop, *faxian* to discover, etc.

[2] 141 sentences were removed because they were in the form of IO V DO.

dom variation in the dataset. In the current study, the datasets in both models contain sentences with different verbs. It is possible that different verbs have different intrinsic tendencies toward a certain word order variant.[3] Incorporating this piece of information into the model makes it more powerful and less affected by the unbalanced distribution of verb types. The mathematical formula of the mixed-effects logistic regression model is given below.

(4) Probability(V DO IO) = $\frac{1}{1+e^{-(\alpha_i + x\beta)}}$, where α_i is the verb-specific intercept of the verb v_i , x is a vector of predictors and β is a vector of corresponding coefficients.

Using the annotated datasets described in 2.2, we built an upper model and a lower model, corresponding to the schema in Fig 2. The general procedure of statistical analysis (which is the same for both models) is described as follows.

We first run the model with all 24 predictors, which will generate a coefficient and a p value for each predictor. Then we refit the model with only significant predictors (i.e. $p < 0.05$). The purpose of doing so is to filter out the noise in the model fit created by the large number of insignificant predictors. Only predictors that remain significant in the simplified model with largely unchanged coefficients are considered to be reliably significant.

Two model evaluation techniques are used to check the model results: cross-validation and separate analysis of high-frequency verbs. A potential problem in any statistical model is that it might overfit the data. After all, what we are interested in is the general probabilistic trends in dative variation, not the trends in a particular set of sentences featuring a particular set of verbs. A cross-validation test helps us evaluate the generalizability of model results by running the same model on a randomly sampled subset of the data. In doing so, it simulates the effect of having different datasets. In practice, we use two types of cross-validation procedures: one randomly samples sentences and the other samples verbs. Each procedure is executed on 100 randomly sampled subset of half the sentences/verbs. Only predictors with consistent performance over all iterations in both tests will be considered as stable.

Another concern in the model design is the effect of verb frequency. In the current dataset, one verb, i.e. *tigong* "to provide", is extremely frequent. 37.3% of the sentences in the upper model and 50.9% in the lower model come from this verb. Though in theory, verb frequency is already taken care of by using mixed-effects models and running cross-validation on samples of the verb set, it is still necessary to test *tigong* separately from the rest of the verbs, due to its extremely high frequency. In the next section, we will report in detail the results from the two regression models.

3 Results

3.1 Upper model: predicting preverbal and postverbal variation

In the upper model, the distinction is between preverbal (DO V IO; coded as 1) and postverbal ditransitives (V DO IO and V IO DO; both coded as 0). The dataset in this model contains 1,149 sentences (of 20 verb types), with 379 preverbal and 770 postverbal. The distribution of the verbs is highly skewed. The most frequent verb is *tigong* "to provide" (n=428 tokens), followed by *song* "to send" (135) and *jiao* "to hand; to transfer" (117). The remaining 17 verbs have between 5 and 54 occurrences in the dataset.

10 out of 24 predictors in the full model are significant and most of them remain significant when the other 14 predictors are removed from the model. Table 1 below summarizes the results of the simplified model.

Judging from the signs of the coefficients in Table 1, a dative sentence is more likely to take the preverbal form (as opposed to the postverbal form) when (a) the verb expresses canonical transfer event, (b) DO is definite, plural, abstract and given in the previous context, with no quantifiers or numerals, (c) IO is not a pronoun and is not given in the previous context, and (d) DO is longer

[3] The same can be said about individual speakers, as some speakers might be more inclined to use certain forms than other speakers. However, since the sentences in the current datasets were sampled from a vast pool of speakers/writers (given the way the corpus is developed), individual differences among speakers is not considered in the current model.

Predictor	β	p
verb is canonical	1.71	0.03
DO is given	1.22	<0.001
DO is definite	4.89	<0.001
DO is plural	1.4	<0.001
DO is concrete	-1.13	0.004
quan/num in DO	-0.99	0.005
IO is pronoun	-1.64	<0.001
IO is given	-0.9	0.007
quan/num in IO	1.32	0.07 (n.s.)
Len(DO)-Len(IO)	0.53	0.002

Table 1: Fixed effects in the simplified upper model

than IO.

Table 2 shows the accuracy of the simplified model. If 0.5 is used as the cut-off probability, the model correctly predicts for (737+338)/1149=93.6% of the sentences. For comparison, the baseline accuracy is only 770/1049=-67% (i.e. by guessing postverbal every time). In other words, the model only needs to include 10 predictors to achieve an increase of around 39% (93.6-67)/67) in model accuracy.

		Predicted	
		preverbal	postverbal
observed	preverbal	338	41
	postverbal	33	737

Table 2: Prediction accuracy of the simplified upper model

Results from the two cross-validation tests confirm all the predictors regarding DO in Table 1, as well as the pronominality of IO and the length difference between DO and IO. Verb category and the givenness of IO do not survive the cross-validation tests.

Separate analysis of tigong shows that indeed, the extremely high-frequency verb exhibits vastly different patterns than other verbs. Only one predictor turns out to be significant for *tigong* sentences, that is, the definiteness of DO (β = 6.17, $p < 0.001$). A closer look at these sentences suggests that they are strongly biased toward postverbal word order, in that 400 out of 428 (95.4%) *tigong* sentences are postverbal (compared with the average level of 67% in all sentences). In other words, just by guessing postverbal every time, one is able to make the correct prediction for *tigong* over 95% of the time. Not surprisingly, there is little need for additional predictors. For non-*tigong* sentences, all factors in Table 1 are significant except for verb category and the presence of quantifiers/numerals in IO. Overall, the non-*tigong* model has an accuracy of 91.5% (baseline = 50.6%).

To sum up, we are confident to say that the semantic features of DO, as well as pronominality of IO and the length difference between the two objects, play important roles in conditioning the preverbal-postverbal variation. Knowing these factors boosts the model s predicting power by a great deal.

3.2 Lower model: predicting pre-IO and post-IO variation

In the lower model, the distinction is between pre-IO sentences (i.e. V DO IO; coded as 1) and post-IO sentences (i.e. V IO DO; coded as 0). The dataset consists of 801 sentences of 14 verb types, among which 161 are pre-IO and 640 are post-IO. The most frequent verb is again, *tigong* (n=408 tokens), followed by *dai* "to bring" (137) and *song* "to send" (89).

Table 3 below summarizes the results of the simplified version of the lower model (constructed in the same fashion as described in Section 3.1).

Predictor	β	p
DO is definite	1.59	0.006
DO is concrete	1.06	<0.001
DO is plural	-0.57	0.04
followed by a verb	2.29	<0.001
normalized verb phrase	1.36	0.13 (n.s.)
Len(DO) - Len(IO)	-1.37	<0.001

Table 3: Fixed effects in the simplified lower model

Compared to the upper model, fewer predictors are significant in the lower model. Everything else

being equal, a postverbal ditransitive sentence is more likely to take the pre-IO form (V DO IO) if (a) DO is definite and concrete, (b) IO is singular, (c) DO is shorter than IO, and (d) the ditransitive construction is followed by another verb. The last point is illustrated in sentence 5a, which is adapted from a real sentence in the corpus. In 5a, the NP *women* "we" is both the recipient of the first verb *song* "to send" and the agent of the second verb *chi* "to eat". Thus, by using a pre-IO form, the NP *women* is in effect adjacent to the second verb chi, which might give an advantage in sentence processing. Notice though, if the other form (V IO DO) is used, the sentence is still grammatical (see 5b).

(5) a. Ta hai song xiaoye gei wo chi.
he also sent snacks to me eat
He also sent snacks for me to eat.

b. Ta hai song (gei) wo xiaoye chi.
he also sent (to) me snacks eat
He also sent me snacks to eat.

Overall the lower model is not as successful as the upper model. The prediction accuracy is 87.7% (baseline accuracy is 79.9%; see Table 4).

		Predicted	
		pre-IO	post-IO
observed	pre-IO	85	76
	post-IO	22	618

Table 4: Prediction accuracy of the simplified lower model

Moreover, cross-validation and the analysis of *tigong* show that only two factors, the presence of the following verb and length difference, are stable across subsets of the data. In fact, with length difference alone, the model generates correct predictions for 86.8% of the sentences (only 1% less than the accuracy reported in Table 4).

However, before we hastily conclude that length difference is the only thing that matters in the lower-level variation, it is important to point out that when the length factor is removed from the model, some predictors (such as the accessibility of DO) turn out to be significant and the model still manages to achieve an accuracy of 85.3%. Therefore, a more plausible explanation is that length difference is the strongest predictors for lower-level dative variation. Though the part of variation it accounts for can also be explained by other predictors, it is more effective in doing so. Therefore the existence of this variable tends to mask other predictors in the model.

4 Discussion

4.1 Comparing he two models

In the current study, we propose a two-level hierarchical schema for modeling the variation among three major word orders of Chinese dative sentences. On the upper level, there is a distinction between sentences with preverbal DOs and those with postverbal DOs. On the lower level, among postverbal sentences, there is a further distinction between pre-IO sentences (i.e. with prepositional phrases), and post-IO sentences (i.e. double object forms). This schema is promoted by structural as well as methodological concerns.

Our modeling results show that the two levels of variation are indeed characterized by different probabilistic patterns, which in turn provide evidence for our original proposal. As presented in Section 3, the upper-level distinction is mostly conditioned by the semantic features of the DO. However, in the lower-level variation, the two best predictors are length difference and the presence of a following verb. Overall, the upper-level model is more successful (accuracy = 93.6%, baseline = 67%) than the lower-level model (accuracy = 87.7%, baseline = 79.9%).

A more striking difference between the two models is that they exhibit weight effects in opposite directions. In both models, length difference between DO and IO plays an important role. Nevertheless, in the upper model, length difference has a positive sign (β = 0.53), meaning that the longer the DO is (compared to the IO), the more likely it is to prepose DO before the verb. Conversely, in the lower-level model, this factor has a negative sign (β = - 1.37), which means that the longer the DO is (compared to the IO), the less likely it is for DO to be before IO. That is to say, everything else being equal, if a DO is long, it will probably be preposed before the verb, but if it is

already after the verb, then it will more likely be placed after IO, at the end of the construction.

The difference in directionality explains why it is only in the lower-level model that the weight effect overshadows other predictors. Features like pronominality, definiteness, and accessibility are inherently correlated with weight. Pronouns are shorter than full NPs; definite NPs tend to be shorter than indefinite NPs (which often take quantifiers and numerals); NPs that have appeared before tend to be in shorter forms than their first occurrences. In both models, a general trend is that NPs that are more prominent in the context (e.g. pronouns, definite NPs, NPs with antecedents) tend to occur earlier in the construction. Thus, in the lower model, the general trend of prominence is confluent with the short before long weight effect, but in the upper model, it is pulling away from the long before short weight effect. As a result, weight effect only masks semantic predictors in the lower model, not in the upper model.

4.2 Comparing with English dative variation

Compared with Bresnan et al.s models, the current results reveal a number of interesting differences between Chinese and English dative variation.

First, the variation phenomenon in Chinese involves at least one more major variant, that is, the preverbal word order, which significantly increases the complexity of the phenomenon. The fact that overall the English model has greater prediction accuracy than the Chinese models might have to do with the fact that the variation phenomenon is more complicated and harder to model in Chinese.

Second, dative variation in Chinese seems to be less sensitive to semantic features. If we only consider the lower-level variation in Chinese, which involves the same form variants as in English (i.e. V DO IO and V IO DO), the Chinese model is best predicted by the length difference between DO and IO and most other predictors are muted by the presence of this factor. In the English model, semantic features are still significant even when length difference is controlled.

Last but not least, as discussed at length in the previous section, the two levels of dative variation in Chinese exhibit weight effects in opposite directions. The English variation is also sensitive to weight, but only in the short before long direction, which is the same as the lower-level variation in Chinese.

5 Conclusion

In this work, we present a corpus-based statistical modeling study on Chinese dative variation. In doing so, we show that this new methodology, which combines corpus data and statistical modeling, is a powerful tool for studying complex variation phenomena in Chinese. The statistical models built in the current study achieve high accuracy in predicting surface forms in Chinese dative sentences. More importantly, the models unveil probabilistic tendencies in Chinese grammar that are otherwise hard to notice.

A remaining question in the current study is *why would Chinese dative variation exhibit weight effects in both directions*. The answer to this question awaits further investigation.

Acknowledgement

We would like to thank three anonymous reviewers for helpful comments on an earlier version of the paper. We owe special thanks to Joan Bresnan and her colleagues in the Spoken Syntax Lab at Stanford University, for sharing working manuals and for valuable discussions.

References

Bresnan, J. (2007). Is syntactic knowledge probabilistic? Experiments with the English dative alternation. In Featherston, S. and Sternefeld, W., editors, *Roots: Linguistics in search of its evidential base*, Studies in generative gramar, pages 77–96. Mouton de Gruyter, Berlin.

Bresnan, J., Cueni, A., Nikitina, T., and Baayen, H. (2007). Predicting the dative alternation. In Boume, G., Kraemer, I., and Zwarts, J., editors, *Cognitive foundations of interpretation*, pages 69–94. Royal Netherlands Academy of Science, Amsterdam.

Gahl, S. and Garnsey, S. (2004). Knowledge of grammar, knowledge of usage: Syntactic prob-

abilities affect pronunciation variation. *Language*, 80(4):748–775.

Huang, C., Chen, K., Chang, L., and Hsu, H. (1995). An introduction to Academia Sinica Balanced Corpus. [in chinese]. In *Proceedings of ROCLING VIII*, pages 81–99.

Jurafsky, D. (2003). Probabilistic modeling in psycholinguistics: Linguistic comprehension and production. In Rens Bod, J. H. and Jannedy, S., editors, *Probabilistic Linguistics*, pages 39–96. MIT Press, Cambridge, Massachusetts.

Levy, R. (2008). Expectation-based syntactic comprehension. *Cognition*, 106(3):1126–1177.

Li, C. N. and Thompson, S. A. (1981). *Mandarin Chinese: A functional reference grammar*. University of California Press, Berkeley.

Liu, F. (2006). Dative constructions in Chinese. *Language and Linguistics*, 7(4):863–904.

Liu, F. H. (2007). Word order variation and ba sentences in Chinese. *Studies in Language*, 31(3):649 – 682.

Manning, C. D. (2003). Probabilistic syntax. In Rens Bod, J. H. and Jannedy, S., editors, *Probabilistic Linguistics*, pages 289–341. MIT Press, Cambridge, Massachusetts.

Margetts, A. and Austin, P. (2007). Three participant events in the languages of the world: toward a cross-linguistic typology. *Linguistics*, 45(3):393–451.

Wasow, T. and Arnold, J. (2003). Post-verbal constituent ordering in english. In Rohdenburg, G. and Mondorf, B., editors, *Determinants of Grammatical Variation in English*, pages 119–154. Mouton.

Appendices

A Complete verb list [4]

song “to send”, *tigong* “to provide”, *jie* “to lend (to)”, *fu* “to pay”, *ban* “to award”, *banfa* “to award”, *zengsong* “to send (as a gift)”, *shang* “to award”, *jieshao* “to introduce”, *huan* “to return”, *fa* “to distribute/allocate”,*jiao* “to transfer”, *ji* “to mail”, *liu* “to leave (behind)”, *liuxia* “to leave (behind)”,*reng* “to throw”, *diu* “to throw”, *diuxia* “to throw (behind)”, *juan* “to donate”, *juanzeng* “to donate”, *juanxian* “to donate”, *bo* “to allocate”, *di* “to hand (to)”, *zu* “to rent (to)”, *fen* “to distribute”, *na* “to hand (to)”, *dai* “to bring”, *dailai* “to bring”, *jiao* “to teach”, *chuan* “to deliver”, *chuanran* “to pass around (a disease)”, *chuanda* “to deliver (a message)”, *chuansong* “to deliver” , *chuanshou* “to deliver (knowledge)”,*ci* “to give (as a reward)”, *pei* “to pay (compensation)”

[4] The verb *gei* “to give” is not included in the list, because it has the same form as the coverb gei and therefore has different properties than other ditransitive verbs. Among other things, the verb *gei* cannot take the V DO IO form in Mandarin (e.g. **gei yiben shu gei wo* “give a book to me”).

B Predictors in the full model

Predictor	**Coding**
genre	1=spoken; 0=written
verb category	1=canonical transfer; 0=otherwise
definiteness of DO	1=definite; 0=indefinite
pronominality of DO	1=pronoun; 0=otherwise
number of DO	1=plural; 0=singular
person of DO	1=1st and 2nd person; 0=otherwise
concreteness of DO	1=concrete; 0=abstract
givenness of DO	1=given; 0=otherwise
quan/num in DO	1=yes; 0=no
definiteness of IO	1=definite; 0=indefinite
pronominality of IO	1=pronoun; 0=otherwise
number of IO	1=plural; 0=singular
person of IO	1=1st and 2nd person; 0=otherwise
concreteness of IO	1=concrete; 0=abstract
givenness of IO	1=given; 0=otherwise
followed by another verb	1=yes; 0=no
embedded under another verb	1=yes; 0=no
part of a copular sentence	1=yes; 0=no
adverbial phrase after the verb	1=yes; 0=no
particle after the verb	1=yes; 0=no
question form	1=yes; 0=no
sentence negation	1=yes; 0=no
relativization	1=yes; 0=no
nominalization	1=yes; 0=no
log(len(DO)- log(len(IO))	numerical

Kernel Slicing: Scalable Online Training with Conjunctive Features

Naoki Yoshinaga
Institute of Industrial Science,
the University of Tokyo
ynaga@tkl.iis.u-tokyo.ac.jp

Masaru Kitsuregawa
Institute of Industrial Science,
the University of Tokyo
kitsure@tkl.iis.u-tokyo.ac.jp

Abstract

This paper proposes an efficient online method that trains a classifier with many conjunctive features. We employ kernel computation called *kernel slicing*, which explicitly considers conjunctions among frequent features in computing the polynomial kernel, to combine the merits of linear and kernel-based training. To improve the scalability of this training, we reuse the temporal margins of partial feature vectors and terminate unnecessary margin computations. Experiments on dependency parsing and hyponymy-relation extraction demonstrated that our method could train a classifier orders of magnitude faster than kernel-based online learning, while retaining its space efficiency.

1 Introduction

The past twenty years have witnessed a growing use of machine-learning classifiers in the field of NLP. Since the classification target of complex NLP tasks (*e.g.*, dependency parsing and relation extraction) consists of more than one constituent (*e.g.*, a head and a dependent in dependency parsing), we need to consider *conjunctive features*, i.e., conjunctions of primitive features that focus on the particular clues of each constituent, to achieve a high degree of accuracy in those tasks.

Training with conjunctive features involves a space-time trade-off in the way conjunctive features are handled. Linear models, such as log-linear models, explicitly estimate the weights of conjunctive features, and training thus requires a great deal of memory when we take higher-order conjunctive features into consideration. Kernel-based models such as support vector machines, on the other hand, ensure space efficiency by using the kernel trick to implicitly consider conjunctive features. However, training takes quadratic time in the number of examples, even with online algorithms such as the (kernel) perceptron (Freund and Schapire, 1999), and we cannot fully exploit ample 'labeled' data obtained with semi-supervised algorithms (Ando and Zhang, 2005; Bellare et al., 2007; Liang et al., 2008; Daumé III, 2008).

We aim at resolving this dilemma in training with conjunctive features, and propose online learning that combines the time efficiency of linear training and the space efficiency of kernel-based training. Following the work by Goldberg and Elhadad (2008), we explicitly take conjunctive features into account that frequently appear in the training data, and implicitly consider the other conjunctive features by using the polynomial kernel. We then improve the scalability of this training by a method called *kernel slicing*, which allows us to reuse the temporal margins of partial feature vectors and to terminate computations that do not contribute to parameter updates.

We evaluate our method in two NLP tasks: dependency parsing and hyponymy-relation extraction. We demonstrate that our method is orders of magnitude faster than kernel-based online learning while retaining its space efficiency.

The remainder of this paper is organized as follows. Section 2 introduces preliminaries and notations. Section 3 proposes our training method. Section 4 evaluates the proposed method. Section 5 discusses related studies. Section 6 concludes this paper and addresses future work.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1245–1253,
Beijing, August 2010

Algorithm 1 BASE LEARNER: KERNEL PA-I

INPUT: $\mathcal{T} = \{(\boldsymbol{x}, y)_t\}_{t=1}^{|\mathcal{T}|}, k : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}, C \in \mathbb{R}^+$
OUTPUT: $(\mathcal{S}_{|\mathcal{T}|}, \boldsymbol{\alpha}_{|\mathcal{T}|})$
1: initialize: $\mathcal{S}_0 \leftarrow \varnothing, \boldsymbol{\alpha}_0 \leftarrow \varnothing$
2: **for** $t = 1$ **to** $|\mathcal{T}|$ **do**
3: receive example $(\boldsymbol{x}, y)_t : \boldsymbol{x} \in \mathbb{R}^n, y \in \{-1, +1\}$
4: compute margin: $m_t(\boldsymbol{x}) = \sum_{\boldsymbol{s}_i \in \mathcal{S}_{t-1}} \alpha_i k(\boldsymbol{s}_i, \boldsymbol{x})$
5: **if** $\ell_t = \max\{0, 1 - y m_t(\boldsymbol{x})\} > 0$ **then**
6: $\tau_t \leftarrow \min\left\{C, \frac{\ell_t}{\|\boldsymbol{x}\|^2}\right\}$
7: $\boldsymbol{\alpha}_t \leftarrow \boldsymbol{\alpha}_{t-1} \cup \{\tau_t y\}, \mathcal{S}_t \leftarrow \mathcal{S}_{t-1} \cup \{\boldsymbol{x}\}$
8: **else**
9: $\boldsymbol{\alpha}_t \leftarrow \boldsymbol{\alpha}_{t-1}, \mathcal{S}_t \leftarrow \mathcal{S}_{t-1}$
10: **end if**
11: **end for**
12: **return** $(\mathcal{S}_{|\mathcal{T}|}, \boldsymbol{\alpha}_{|\mathcal{T}|})$

2 Preliminaries

This section first introduces a passive-aggressive algorithm (Crammer et al., 2006), which we use as a base learner. We then explain fast methods of computing the polynomial kernel.

Each example $\boldsymbol{x}$ in a classification problem is represented by a *feature vector* whose element x_j is a value of a feature function, $f_j \in \mathcal{F}$. Here, we assume a binary feature function, $f_j(\boldsymbol{x}) \in \{0, 1\}$, which returns one if particular context data appear in the example. We say that feature f_j is *active* in example $\boldsymbol{x}$ when $x_j = f_j(\boldsymbol{x}) = 1$. We denote a binary feature vector, $\boldsymbol{x}$, as a set of active features $\boldsymbol{x} = \{f_j \mid f_j \in \mathcal{F}, f_j(\boldsymbol{x}) = 1\}$ for brevity; $f_j \in \boldsymbol{x}$ means that f_j is active in $\boldsymbol{x}$, and $|\boldsymbol{x}|$ represents the number of active features in $\boldsymbol{x}$.

2.1 Kernel Passive-Aggressive Algorithm

A passive-aggressive algorithm (PA) (Crammer et al., 2006) represents online learning that updates parameters for given labeled example $(\boldsymbol{x}, y)_t \in \mathcal{T}$ in each round t. We assume a binary label, $y \in \{-1, +1\}$, here for clarity. Algorithm 1 is a variant of PA (PA-I) that incorporates a kernel function, k. In round t, PA-I first computes a *(signed) margin* $m_t(\boldsymbol{x})$ of $\boldsymbol{x}$ by using the kernel function with support set $\mathcal{S}_{t-1}$ and coefficients $\boldsymbol{\alpha}_{t-1}$ (Line 4). PA-I then suffers a hinge-loss, $\ell_t = \max\{0, 1 - y m_t(\boldsymbol{x})\}$ (Line 5). If $\ell_t > 0$, PA-I adds $\boldsymbol{x}$ to $\mathcal{S}_{t-1}$ (Line 7). Hyperparameter C controls the aggressiveness of parameter updates.

The kernel function computes a dot product in $\mathbb{R}^{\mathcal{H}}$ space without mapping $\boldsymbol{x} \in \mathbb{R}^n$ to $\boldsymbol{\phi}(\boldsymbol{x}) \in \mathbb{R}^{\mathcal{H}}$ ($k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}')$). We can implicitly consider (weighted) d or less order conjunctions of primitive features by using *polynomial kernel function* $k_d(\boldsymbol{s}, \boldsymbol{x}) = (\boldsymbol{s}^{\mathrm{T}} \boldsymbol{x} + 1)^d$. For example, given support vector $\boldsymbol{s} = (s_1, s_2)^{\mathrm{T}}$ and input example $\boldsymbol{x} = (x_1, x_2)^{\mathrm{T}}$, the second-order polynomial kernel returns $k_2(\boldsymbol{s}, \boldsymbol{x}) = (s_1 x_1 + s_2 x_2 + 1)^2 = 1 + 3 s_1 x_1 + 3 s_2 x_2 + 2 s_1 x_1 s_2 x_2$ ($\because s_i, x_i \in \{0, 1\}$). This function thus implies mapping $\boldsymbol{\phi}_2(\boldsymbol{x}) = (1, \sqrt{3} x_1, \sqrt{3} x_2, \sqrt{2} x_1 x_2)^{\mathrm{T}}$.

Although online learning is generally efficient, the kernel spoils its efficiency (Dekel et al., 2008). This is because the kernel evaluation (Line 4) takes $\mathcal{O}(|\mathcal{S}_{t-1}||\boldsymbol{x}|)$ time and $|\mathcal{S}_{t-1}|$ increases as training continues. The learner thus takes the most amount of time in this margin computation.

2.2 Kernel Computation for Classification

This section explains fast, exact methods of computing the polynomial kernel, which are meant to test the trained model, $(\mathcal{S}, \boldsymbol{\alpha})$, and involve substantial computational cost in preparation.

2.2.1 Kernel Inverted

Kudo and Matsumoto (2003) proposed *polynomial kernel inverted* (PKI), which builds inverted indices $h(f_j) \equiv \{\boldsymbol{s} \mid \boldsymbol{s} \in \mathcal{S}, f_j \in \boldsymbol{s}\}$ from each feature f_j to support vector $\boldsymbol{s} \in \mathcal{S}$ to only consider support vector $\boldsymbol{s}$ relevant to given $\boldsymbol{x}$ such that $\boldsymbol{s}^{\mathrm{T}} \boldsymbol{x} \neq 0$. The time complexity of PKI is $\mathcal{O}(B \cdot |\boldsymbol{x}| + |\mathcal{S}|)$ where $B \equiv \frac{1}{|\boldsymbol{x}|} \sum_{f_j \in \boldsymbol{x}} |h(f_j)|$, which is smaller than $\mathcal{O}(|\mathcal{S}||\boldsymbol{x}|)$ if $\boldsymbol{x}$ has many rare features f_j such that $|h(f_j)| \ll |\mathcal{S}|$.

To the best of our knowledge, this is the only exact method that has been used to speed up margin computation in the context of kernel-based online learning (Okanohara and Tsujii, 2007).

2.2.2 Kernel Expansion

Isozaki and Kazawa (2002) and Kudo and Matsumoto (2003) proposed *kernel expansion*, which explicitly maps both support set $\mathcal{S}$ and given example $\boldsymbol{x} \in \mathbb{R}^n$ into $\mathbb{R}^{\mathcal{H}}$ by mapping $\boldsymbol{\phi}_d$ imposed by k_d:

$$m(\boldsymbol{x}) = \left(\sum_{\boldsymbol{s}_i \in \mathcal{S}} \alpha_i \boldsymbol{\phi}_d(\boldsymbol{s}_i) \right)^{\mathrm{T}} \boldsymbol{\phi}_d(\boldsymbol{x}) = \sum_{f_i \in \boldsymbol{x}^d} w_i,$$

where $\boldsymbol{x}^d \in \{0,1\}^{\mathcal{H}}$ is a binary feature vector in which $x_i^d = 1$ for $(\boldsymbol{\phi}_d(\boldsymbol{x}))_i \neq 0$, and $\boldsymbol{w}$ is a weight vector in the expanded feature space, $\mathcal{F}^d$. The weight vector $\boldsymbol{w}$ is computed from $\mathcal{S}$ and $\boldsymbol{\alpha}$:

$$\boldsymbol{w} = \sum_{\boldsymbol{s}_i \in \mathcal{S}} \alpha_i \sum_{k=0}^{d} c_d^k I_k(\boldsymbol{s}_i^d), \quad (1)$$

where c_d^k is a squared coefficient of k-th order conjunctive features for d-th order polynomial kernel (*e.g.*, $c_2^0 = 1$, $c_2^1 = 3$, and $c_2^2 = 2$)[1] and $I_k(\boldsymbol{s}_i^d)$ is $\boldsymbol{s}_i^d \in \{0,1\}^{\mathcal{H}}$ whose dimensions other than those of k-th order conjunctive features are set to zero.

The time complexity of kernel expansion is $\mathcal{O}(|\boldsymbol{x}^d|)$ where $|\boldsymbol{x}^d| = \sum_{k=0}^{d} \binom{|\boldsymbol{x}|}{k} \propto |\boldsymbol{x}|^d$, which can be smaller than $\mathcal{O}(|\mathcal{S}||\boldsymbol{x}|)$ in usual NLP tasks ($|\boldsymbol{x}| \ll |\mathcal{S}|$ and $d \leq 4$).

2.2.3 Kernel Splitting

Since kernel expansion demands a huge memory volume to store the weight vector, $\boldsymbol{w}$, in $\mathbb{R}^{\mathcal{H}}$ ($\mathcal{H} = \sum_{k=0}^{d} \binom{|\mathcal{F}|}{k}$), Goldberg and Elhadad (2008) only explicitly considered conjunctions among features $f_C \in \mathcal{F}_C$ that commonly appear in support set $\mathcal{S}$, and handled the other conjunctive features relevant to rare features $f_R \in \mathcal{F} \setminus \mathcal{F}_C$ by using the polynomial kernel:

$$\begin{aligned} m(\boldsymbol{x}) &= m(\tilde{\boldsymbol{x}}) + m(\boldsymbol{x}) - m(\tilde{\boldsymbol{x}}) \\ &= \sum_{f_i \in \tilde{\boldsymbol{x}}^d} \tilde{w}_i + \sum_{\boldsymbol{s}_i \in \mathcal{S}_R} \alpha_i k'_d(\boldsymbol{s}_i, \boldsymbol{x}, \tilde{\boldsymbol{x}}), \quad (2) \end{aligned}$$

where $\tilde{\boldsymbol{x}}$ is $\boldsymbol{x}$ whose dimensions of rare features are set to zero, $\tilde{\boldsymbol{w}}$ is a weight vector computed with Eq. 1 for $\mathcal{F}_C^d$, and $k'_d(\boldsymbol{s}, \boldsymbol{x}, \tilde{\boldsymbol{x}})$ is defined as:

$$k'_d(\boldsymbol{s}, \boldsymbol{x}, \tilde{\boldsymbol{x}}) \equiv k_d(\boldsymbol{s}, \boldsymbol{x}) - k_d(\boldsymbol{s}, \tilde{\boldsymbol{x}}).$$

We can space-efficiently compute the first term of Eq. 2 since $|\tilde{\boldsymbol{w}}| \ll |\boldsymbol{w}|$, while we can quickly compute the second term of Eq. 2 since $k'_d(\boldsymbol{s}_i, \boldsymbol{x}, \tilde{\boldsymbol{x}}) = 0$ when $\boldsymbol{s}_i^{\mathrm{T}} \boldsymbol{x} = \boldsymbol{s}_i^{\mathrm{T}} \tilde{\boldsymbol{x}}$; we only need to consider a small subset of the support set, $\mathcal{S}_R = \bigcup_{f_R \in \boldsymbol{x} \setminus \tilde{\boldsymbol{x}}} h(f_R)$, that has at least one of the rare features, f_R, appearing in $\boldsymbol{x} \setminus \tilde{\boldsymbol{x}}$ ($|\mathcal{S}_R| \ll |\mathcal{S}|$).

Counting the number of features examined, the time complexity of Eq. 2 is $\mathcal{O}(|\tilde{\boldsymbol{x}}^d| + |\mathcal{S}_R||\tilde{\boldsymbol{x}}|)$.

[1] Following Lemma 1 in Kudo and Matsumoto (2003), $c_d^k = \sum_{l=k}^{d} \binom{d}{l} \left(\sum_{m=0}^{k} (-1)^{k-m} \cdot m^l \binom{k}{m}\right)$.

3 Algorithm

This section first describes the way kernel splitting is integrated into PA-I (Section 3.1). We then propose *kernel slicing* (Section 3.2), which enables us to reuse the temporal margins computed in the past rounds (Section 3.2.1) and to skip unnecessary margin computations (Section 3.2.2).

In what follows, we use PA-I as a base learner. Note that an analogous argument can be applied to other perceptron-like online learners with the additive weight update (Line 7 in Algorithm 1).

3.1 Base Learner with Kernel Splitting

A problem in integrating kernel splitting into the base learner presented in Algorithm 1 is how to determine $\mathcal{F}_C$, features among which we explicitly consider conjunctions, without knowing the final support set, $\mathcal{S}_{|\mathcal{T}|}$. We heuristically solve this by ranking feature f according to their frequency in the training data and by using the top-N frequent features in the training data as $\mathcal{F}_C$ ($= \{f \mid f \in \mathcal{F}, \mathrm{RANK}(f) \leq N\}$).[2] Since $\mathcal{S}_{|\mathcal{T}|}$ is a subset of the examples, this approximates the selection from $\mathcal{S}_{|\mathcal{T}|}$. We empirically demonstrate the validity of this approach in the experiments.

We then use $\mathcal{F}_C$ to construct a base learner with kernel splitting; we replace the kernel computation (Line 4 in Algorithm 1) with Eq. 2 where $(\mathcal{S}, \boldsymbol{\alpha}) = (\mathcal{S}_{t-1}, \boldsymbol{\alpha}_{t-1})$. To compute $m_t(\tilde{\boldsymbol{x}})$ by using kernel expansion, we need to additionally maintain the weight vector $\tilde{\boldsymbol{w}}$ for the conjunctions of common features that appear in $\mathcal{S}_{t-1}$.

The additive parameter update of PA-I enables us to keep $\tilde{\boldsymbol{w}}$ to correspond to $(\mathcal{S}_{t-1}, \boldsymbol{\alpha}_{t-1})$. When we add $\boldsymbol{x}$ to support set $\mathcal{S}_{t-1}$ (Line 7 in Algorithm 1), we also update $\tilde{\boldsymbol{w}}$ with Eq. 1:

$$\tilde{\boldsymbol{w}} \leftarrow \tilde{\boldsymbol{w}} + \tau_t y \sum_{k=0}^{d} c_d^k I_k(\tilde{\boldsymbol{x}}^d).$$

Following (Kudo and Matsumoto, 2003), we use a trie (hereafter, *weight trie*) to maintain conjunctive features. Each edge in the weight trie is labeled with a primitive feature, while each path

[2] The overhead of counting features is negligible compared to the total training time. If we want to run the learner in a purely online manner, we can alternatively choose first N features that appear in the processed examples as $\mathcal{F}_C$.

represents a conjunctive feature that combines all the primitive features on the path. The weights of conjunctive features are retrieved by traversing nodes in the trie. We carry out an analogous traversal in updating the parameters of conjunctive features, while registering a new conjunctive feature by adding an edge to the trie.

The base learner with kernel splitting combines the virtues of linear training and kernel-based training. It reduces to linear training when we increase N to $|\mathcal{F}|$, while it reduces to kernel-based training when we decrease N to 0. The output is support set $\mathcal{S}_{|\mathcal{T}|}$ and coefficients $\boldsymbol{\alpha}_{|\mathcal{T}|}$ (optionally, $\tilde{\boldsymbol{w}}$), to which the efficient classification techniques discussed in Section 2.2 and the one proposed by Yoshinaga and Kitsuregawa (2009) can be applied.

Note on weight trie construction The time and space efficiency of this learner strongly depends on the way the weight trie is constructed. We need to address two practical issues that greatly affect efficiency. First, we traverse the trie from the rarest feature that constitutes a conjunctive feature. This rare-to-frequent mining helps us to avoid enumerating higher-order conjunctive features that have not been registered in the trie, when computing margin. Second, we use $\textsc{rank}(f)$ encoded into a $\lceil \log_{128} \textsc{rank}(f) \rceil$-byte string by using variable-byte coding (Williams and Zobel, 1999) as f's representation in the trie. This encoding reduces the trie size, since features with small $\textsc{rank}(f)$ will appear frequently in the trie.

3.2 Base Learner with Kernel Slicing

Although a base learner with kernel splitting can enjoy the merits of linear and kernel-based training, it can simultaneously suffer from their demerits. Because the training takes polynomial time in the number of common features in $\boldsymbol{x}$ ($|\tilde{\boldsymbol{x}}^d| = \sum_{k=0}^{d} \binom{|\tilde{\boldsymbol{x}}|}{k} \propto |\tilde{\boldsymbol{x}}|^d$) at each round, we need to set N to a smaller value when we take higher-order conjunctive features into consideration. However, since the margin computation takes linear time in the number of support vectors $|\mathcal{S}_R|$ relevant to rare features $f_R \in \mathcal{F} \setminus \mathcal{F}_C$, we need to set N to a larger value when we handle a larger number of training examples. The training thereby slows down when we train a classifier with high-order conjunctive features and a large number of training examples.

We then attempt to improve the scalability of the training by exploiting a characteristic of labeled data in NLP. Because examples in NLP tasks are likely to be redundant (Yoshinaga and Kitsuregawa, 2009), the learner computes margins of examples that have many features in common. If we can reuse the 'temporal' margins of partial feature vectors computed in past rounds, this will speed up the computation of margins.

We propose *kernel slicing*, which generalizes kernel splitting in a purely feature-wise manner and enables us to reuse the temporal partial margins. Starting from the most frequent feature f_1 in $\boldsymbol{x}$ ($f_1 = \text{argmin}_{f \in \boldsymbol{x}} \textsc{rank}(f)$), we incrementally compute $m_t(\boldsymbol{x})$ by accumulating a *partial margin*, $m_t^j(\boldsymbol{x}) \equiv m_t(\boldsymbol{x}_j) - m_t(\boldsymbol{x}_{j-1})$, when we add the j-th frequent feature f_j in $\boldsymbol{x}$:

$$m_t(\boldsymbol{x}) = m_t^0 + \sum_{j=1}^{|\boldsymbol{x}|} m_t^j(\boldsymbol{x}), \tag{3}$$

where $m_t^0 = \sum_{\boldsymbol{s}_i \in \mathcal{S}_{t-1}} \alpha_i k_d(\boldsymbol{s}_i, \varnothing) = \sum_i \alpha_i$, and $\boldsymbol{x}_j$ has the j most frequent features in $\boldsymbol{x}$ ($\boldsymbol{x}_0 = \varnothing$, $\boldsymbol{x}_j = \bigsqcup_{k=0}^{j-1} \{\text{argmin}_{f \in \boldsymbol{x} \setminus \boldsymbol{x}_k} \textsc{rank}(f)\}$).

Partial margin $m_t^j(\boldsymbol{x})$ can be computed by using the polynomial kernel:

$$m_t^j(\boldsymbol{x}) = \sum_{\boldsymbol{s}_i \in \mathcal{S}_{t-1}} \alpha_i k'_d(\boldsymbol{s}_i, \boldsymbol{x}_j, \boldsymbol{x}_{j-1}), \tag{4}$$

or by using kernel expansion:

$$m_t^j(\boldsymbol{x}) = \sum_{f_i \in \boldsymbol{x}_j^d \setminus \boldsymbol{x}_{j-1}^d} \tilde{w}_i. \tag{5}$$

Kernel splitting is a special case of kernel slicing, which uses Eq. 5 for $f_j \in \mathcal{F}_C$ and Eq. 4 for $f_j \in \mathcal{F} \setminus \mathcal{F}_C$.

3.2.1 Reuse of Temporal Partial Margins

We can speed up both Eqs. 4 and 5 by reusing a temporal partial margin, $\delta_{t'}^j = m_{t'}^j(\boldsymbol{x})$ that had been computed in past round $t' (< t)$:

$$m_t^j(\boldsymbol{x}) = \delta_{t'}^j + \sum_{\boldsymbol{s}_i \in \mathcal{S}_j} \alpha_i k'_d(\boldsymbol{s}_i, \boldsymbol{x}_j, \boldsymbol{x}_{j-1}), \tag{6}$$

where $\mathcal{S}_j = \{\boldsymbol{s} \mid \boldsymbol{s} \in \mathcal{S}_{t-1} \setminus \mathcal{S}_{t'-1}, f_j \in \boldsymbol{s}\}$.

Algorithm 2 KERNEL SLICING

INPUT: $\boldsymbol{x} \in 2^{\mathcal{F}}, \mathcal{S}_{t-1}, \boldsymbol{\alpha}_{t-1}, \mathcal{F}_C \subseteq \mathcal{F}, \boldsymbol{\delta} : 2^{\mathcal{F}} \mapsto \mathbb{N} \times \mathbb{R}$
OUTPUT: $m_t(\boldsymbol{x})$

1: initialize: $\boldsymbol{x}_0 \leftarrow \varnothing, j \leftarrow 1, m_t(\boldsymbol{x}) \leftarrow m_t^0$
2: **repeat**
3: $\quad \boldsymbol{x}_j \leftarrow \boldsymbol{x}_{j-1} \sqcup \{\mathrm{argmin}_{f \in \boldsymbol{x} \setminus \boldsymbol{x}_{j-1}} \textsc{rank}(f)\}$
4: $\quad$ retrieve partial margin: $(t', \delta_{t'}^j) \leftarrow \boldsymbol{\delta}(\boldsymbol{x}_j)$
5: $\quad$ **if** $f_j \in \mathcal{F} \setminus \mathcal{F}_C$ **or** Eq. 7 is **true then**
6: $\quad\quad$ compute $m_t^j(\boldsymbol{x})$ using Eq. 6 with $\delta_{t'}^j$
7: $\quad$ **else**
8: $\quad\quad$ compute $m_t^j(\boldsymbol{x})$ using Eq. 5
9: $\quad$ **end if**
10: $\quad$ update partial margin: $\boldsymbol{\delta}(\boldsymbol{x}_j) \leftarrow (t, m_t^j(\boldsymbol{x}))$
11: $\quad m_t(\boldsymbol{x}) \leftarrow m_t(\boldsymbol{x}) + m_t^j(\boldsymbol{x})$
12: **until** $\boldsymbol{x}_j \neq \boldsymbol{x}$
13: **return** $m_t(\boldsymbol{x})$

Eq. 6 is faster than Eq. 4,[3] and can even be faster than Eq. 5.[4] When $\textsc{rank}(f_j)$ is high, $\boldsymbol{x}_j$ appears frequently in the training examples and $|\mathcal{S}_j|$ becomes small since t' will be close to t. When $\textsc{rank}(f_j)$ is low, $\boldsymbol{x}_j$ rarely appears in the training examples but we can still expect $|\mathcal{S}_j|$ to be small since the number of support vectors in $\mathcal{S}_{t-1} \setminus \mathcal{S}_{t'-1}$ that have rare feature f_j will be small.

To compute Eq. 3, we now have the choice to choose Eq. 5 or 6 for $f_j \in \mathcal{F}_C$. Counting the number of features to be examined in computing $m_t^j(\boldsymbol{x})$, we have the following criteria to determine whether we can use Eq. 6 instead of Eq. 5:

$$1 + |\mathcal{S}_j||\boldsymbol{x}_{j-1}| \leq |\boldsymbol{x}_j^d \setminus \boldsymbol{x}_{j-1}^d| = \sum_{k=1}^{d} \binom{j-1}{k-1},$$

where the left- and right-hand sides indicate the number of features examined in Eq. 6 for the former and Eq. 5 for the latter. Expanding the right-hand side for $d = 2, 3$ and dividing both sides with $|\boldsymbol{x}_{j-1}| = j - 1$, we have:

$$|\mathcal{S}_j| \leq \begin{cases} 1 & (d = 2) \\ \frac{j}{2} & (d = 3) \end{cases}. \tag{7}$$

If this condition is met after retrieving the temporal partial margin, $\delta_{t'}^j$, we can compute partial margin $m_t^j(\boldsymbol{x})$ with Eq. 6. This analysis reveals that we can expect little speed-up for the second-order polynomial kernel; we will only use Eq. 6 with third or higher-order polynomial kernel.

Algorithm 2 summarizes the margin computation with kernel slicing. It processes each feature $f_j \in \boldsymbol{x}$ in frequent-to-rare order, and accumulates partial margin $m_t^j(\boldsymbol{x})$ to have $m_t(\boldsymbol{x})$. Intuitively speaking, when the algorithm uses the partial margin, it only considers support vectors on each feature that have been added since the last evaluation of the partial feature vector, to avoid the repetition in kernel evaluation as much as possible.

3.2.2 Termination of Margin Computation

Kernel slicing enables another optimization that exploits a characteristic of online learning. Because we need an exact margin, $m_t(\boldsymbol{x})$, only when hinge-loss $\ell_t = 1 - y m_t(\boldsymbol{x})$ is positive, we can finish margin computation as soon as we find that the lower-bound of $y m_t(\boldsymbol{x})$ is larger than one.

When $y m_t(\boldsymbol{x})$ is larger than one after processing feature f_j in Eq. 3, we quickly examine whether this will hold even after we process the remaining features. We can compute a possible range of partial margin $m_t^k(\boldsymbol{x})$ with Eq. 4, having the upper- and lower-bounds, $\hat{k}'_d$ and $\check{k}'_d$, of $k'_d(\boldsymbol{s}_i, \boldsymbol{x}_k, \boldsymbol{x}_{k-1})$ $(= k_d(\boldsymbol{s}_i, \boldsymbol{x}_k) - k_d(\boldsymbol{s}_i, \boldsymbol{x}_{k-1}))$:

$$m_t^k(\boldsymbol{x}) \leq \hat{k}'_d \sum_{\boldsymbol{s}_i \in \mathcal{S}_k^+} \alpha_i + \check{k}'_d \sum_{\boldsymbol{s}_i \in \mathcal{S}_k^-} \alpha_i \tag{8}$$

$$m_t^k(\boldsymbol{x}) \geq \check{k}'_d \sum_{\boldsymbol{s}_i \in \mathcal{S}_k^+} \alpha_i + \hat{k}'_d \sum_{\boldsymbol{s}_i \in \mathcal{S}_k^-} \alpha_i, \tag{9}$$

where $\mathcal{S}_k^+ = \{\boldsymbol{s}_i \,|\, \boldsymbol{s}_i \in \mathcal{S}_{t-1}, f_k \in \boldsymbol{s}_i, \alpha_i > 0\}$, $\mathcal{S}_k^- = \{\boldsymbol{s}_i \,|\, \boldsymbol{s}_i \in \mathcal{S}_{t-1}, f_k \in \boldsymbol{s}_i, \alpha_i < 0\}$, $\hat{k}'_d = (k+1)^d - k^d$ and $\check{k}'_d = 2^d - 1$ $(\because 0 \leq \boldsymbol{s}_i^{\mathrm{T}} \boldsymbol{x}_{k-1} \leq |\boldsymbol{x}_{k-1}| = k - 1,\ \boldsymbol{s}_i^{\mathrm{T}} \boldsymbol{x}_k = \boldsymbol{s}_i^{\mathrm{T}} \boldsymbol{x}_{k-1} + 1$ for all $\boldsymbol{s}_i \in \mathcal{S}_k^+ \cup \mathcal{S}_k^-)$.

We accumulate Eqs. 8 and 9 from rare to frequent features, and use the intermediate results to estimate the possible range of $m_t(\boldsymbol{x})$ before Line 3 in Algorithm 2. If the lower bound of $y m_t(\boldsymbol{x})$ turns out to be larger than one, we terminate the computation of $m_t(\boldsymbol{x})$.

As training continues, the model becomes discriminative and given $\boldsymbol{x}$ is likely to have a larger margin. The impact of this termination will increase as the amount of training data expands.

[3] When a margin of $\boldsymbol{x}_j$ has not been computed, we regard $t' = 0$ and $\delta_{t'}^j = 0$, which reduces Eq. 6 to Eq. 4.

[4] We associate partial margins with partial feature sequences whose features are sorted by frequent-to-rare order, and store them in a trie (*partial margin trie*). This enables us to retrieve partial margin $\delta_{t'}^j$ for given $\boldsymbol{x}_j$ in $\mathcal{O}(1)$ time.

4 Evaluation

We evaluated the proposed method in two NLP tasks: dependency parsing (Sassano, 2004) and hyponymy-relation extraction (Sumida et al., 2008). We used labeled data included in open-source softwares to promote the reproducibility of our results.[5] All the experiments were conducted on a server with an Intel® Xeon™ 3.2 GHz CPU. We used a double-array trie (Aoe, 1989; Yata et al., 2009) as an implementation of the weight trie and the partial margin trie.

4.1 Task Descriptions

Japanese Dependency Parsing A parser inputs a sentence segmented by a *bunsetsu* (base phrase in Japanese), and selects a particular pair of bunsetsus (dependent and head candidates); the classifier then outputs label $y = +1$ (dependent) or -1 (independent) for the pair. The features consist of the surface form, POS, POS-subcategory and the inflection form of each bunsetsu, and surrounding contexts such as the positional distance, punctuations and brackets. See (Yoshinaga and Kitsuregawa, 2009) for details on the features.

Hyponymy-Relation Extraction A hyponymy relation extractor (Sumida et al., 2008) first extracts a pair of entities from hierarchical listing structures in Wikipedia articles (hypernym and hyponym candidates); a classifier then outputs label $y = +1$ (correct) or -1 (incorrect) for the pair. The features include a surface form, morphemes, POS and the listing type for each entity, and surrounding contexts such as the hierarchical distance between the entities. See (Sumida et al., 2008) for details on the features.

4.2 Settings

Table 1 summarizes the training data for the two tasks. The examples for the Japanese dependency parsing task were generated for a transition-based parser (Sassano, 2004) from a standard data set.[6] We used the dependency accuracy of the parser as model accuracy in this task. In the hyponymy-relation extraction task, we randomly chosen two sets of 10,000 examples from the labeled data for development and testing, and used the remaining examples for training. Note that the number of active features, $|\mathcal{F}^d|$, dramatically grows when we consider higher-order conjunctive features.

DATA SET	DEP	REL
$\|\mathcal{T}\|$	296,776	201,664
$(y = +1)$	150,064	152,199
$(y = -1)$	146,712	49,465
Ave. of $\|\boldsymbol{x}\|$	27.6	15.4
Ave. of $\|\boldsymbol{x}^2\|$	396.1	136.9
Ave. of $\|\boldsymbol{x}^3\|$	3558.3	798.7
$\|\mathcal{F}\|$	64,493	306,036
$\|\mathcal{F}^2\|$	3,093,768	6,688,886
$\|\mathcal{F}^3\|$	58,361,669	64,249,234

Table 1: Training data for dependency parsing (DEP) and hyponymy-relation extraction (REL).

We compared the proposed method, PA-I SL (Algorithm 1 with Algorithm 2), to PA-I KERNEL (Algorithm 1 with PKI; Okanohara and Tsujii (2007)), PA-I KE (Algorithm 1 with kernel expansion; viz., kernel splitting with $N = |\mathcal{F}|$), SVM (batch training of support vector machines),[7] and ℓ_1-LLM (stochastic gradient descent training of the ℓ_1-regularized log-linear model: Tsuruoka et al. (2009)). We refer to PA-I SL that does not reuse temporal partial margins as PA-I SL*. To demonstrate the impact of conjunctive features on model accuracy, we also trained PA-I without conjunctive features. The number of iterations in PA-I was set to 20, and the parameters of PA-I were averaged in an efficient manner (Daumé III, 2006). We explicitly considered conjunctions among top-N ($N = 125 \times 2^n; n \geq 0$) features in PA-I SL and PA-I SL*. The hyperparameters were tuned to maximize accuracy on the development set.

4.3 Results

Tables 2 and 3 list the experimental results for the two tasks (due to space limitations, Tables 2 and 3 list PA-I SL with parameter N that achieved the fastest speed). The accuracy of the models trained with the proposed method was better than ℓ_1-LLMs and was comparable to SVMs. The infe-

[5] The labeled data for dependency parsing is available from: http://www.tkl.iis.u-tokyo.ac.jp/~ynaga/pecco/, and the labeled data for hyponymy-relation extraction is available from: http://nlpwww.nict.go.jp/hyponymy/.

[6] Kyoto Text Corpus Version 4.0: http://nlp.kuee.kyoto-u.ac.jp/nl-resource/corpus-e.html.

[7] http://chasen.org/~taku/software/TinySVM/

METHOD	d	ACC.	TIME	MEMORY
PA-I	1	88.56%	3s	55MB
ℓ_1-LLM	2	90.55%	340s	1656MB
SVM	2	90.76%	29863s	245MB
PA-I KERNEL	2	90.68%	8361s	84MB
PA-I KE	2	90.67%	41s	155MB
PA-I SL$^*_{N=4000}$	2	90.71%	**33s**	95MB
ℓ_1-LLM	3	90.76%	4057s	21,499MB
SVM	3	90.93%	25912s	243MB
PA-I KERNEL	3	90.90%	8704s	83MB
PA-I KE	3	90.90%	465s	993MB
PA-I SL$_{N=250}$	3	90.89%	**262s**	175MB

Table 2: Training time for classifiers used in dependency parsing task.

METHOD	d	ACC.	TIME	MEMORY
PA-I	1	91.75%	2s	28MB
ℓ_1-LLM	2	92.67%	136s	1683MB
SVM	2	92.85%	12306s	139MB
PA-I KERNEL	2	92.91%	1251s	54MB
PA-I KE	2	92.96%	27s	143MB
PA-I SL$^*_{N=8000}$	2	92.88%	**17s**	77MB
ℓ_1-LLM	3	92.86%	779s	14,089MB
SVM	3	93.09%	17354s	140MB
PA-I KERNEL	3	93.14%	1074s	49MB
PA-I KE	3	93.11%	103s	751MB
PA-I SL$_{N=125}$	3	93.05%	**17s**	131MB

Table 3: Training time for classifiers used in hyponymy-relation extraction task.

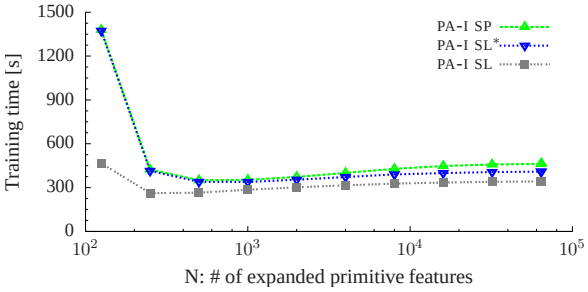


Figure 1: Training time for PA-I variants as a function of the number of expanded primitive features in dependency parsing task ($d = 3$).

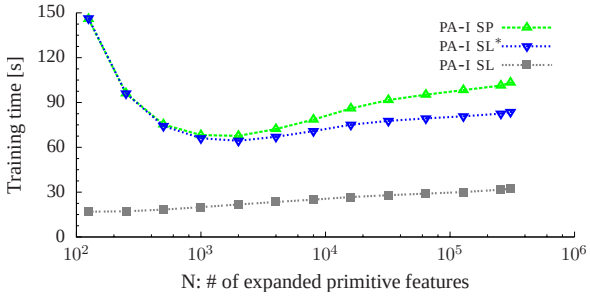


Figure 2: Training time for PA-I variants as a function of the number of expanded primitive features in hyponymy-relation extraction task ($d = 3$).

rior accuracy of PA-I ($d = 1$) confirmed the necessity of conjunctive features in these tasks. The minor difference among the model accuracy of the three PA-I variants was due to rounding errors.

PA-I SL was the fastest of the training methods with the same feature set, and its space efficiency was comparable to the kernel-based learners. PA-I SL could reduce the memory footprint from 993MB[8] to 175MB for $d = 3$ in the dependency parsing task, while speeding up training.

Although linear training (ℓ_1-LLM and PA-I KE) dramatically slowed down when we took higher-order conjunctive features into account, kernel slicing alleviated deterioration in speed. Especially in the hyponymy-relation extraction task, PA-I SL took almost the same time regardless of the order of conjunctive features.

[8] ℓ_1-LLM took much more memory than PA-I KE mainly because ℓ_1-LLM expands conjunctive features in the examples prior to training, while PA-I KE expands conjunctive features in each example on the fly during training. Interested readers may refer to (Chang et al., 2010) for this issue.

Figures 1 and 2 plot the trade-off between the number of expanded primitive features and training time with PA-I variants ($d = 3$) in the two tasks. Here, PA-I SP is PA-I with kernel slicing without the techniques described in Sections 3.2.1 and 3.2.2, viz., kernel splitting. The early termination of margin computation reduces the training time when N is large. The reuse of temporal margins makes the training time stable regardless of parameter N. This suggests a simple, effective strategy for calibrating N; we start the training with $N = |\mathcal{F}|$, and when the learner reaches the allowed memory size, we shrink N to $N/2$ by pruning sub-trees rooted by rarer features with $\text{RANK}(f) > N/2$ in the weight trie.

Figures 3 and 4 plot training time with PA-I variants ($d = 3$) for the two tasks as a function of the training data size. PA-I SP inherited the demerit of PA-I KERNEL which takes quadratic time in the number of examples, while PA-I SL took almost linear time in the number of examples.

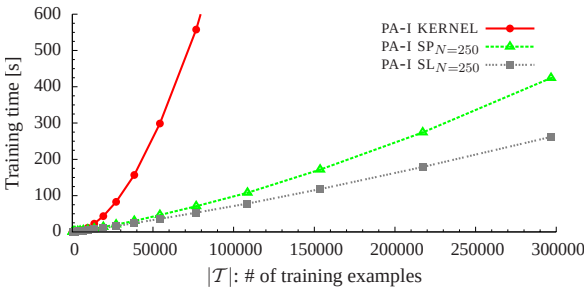


Figure 3: Training time for PA-I variants as a function of the number of training examples in dependency parsing task ($d = 3$).

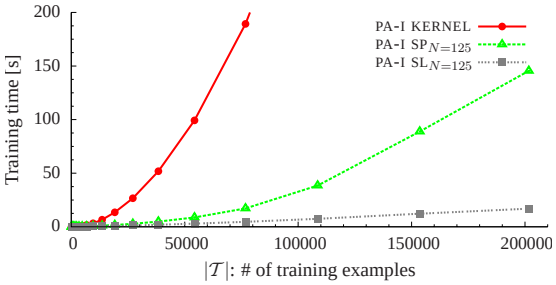


Figure 4: Training time for PA-I variants as a function of the number of training examples in hyponymy-relation extraction task ($d = 3$).

5 Related Work

There are several methods that learn 'simpler' models with fewer variables (features or support vectors), to ensure scalability in training.

Researchers have employed feature selection to assure space-efficiency in linear training. Wu et al. (2007) used frequent-pattern mining to select effective conjunctive features prior to training. Okanohara and Tsujii (2009) revised grafting for ℓ_1-LLM (Perkins et al., 2003) to prune useless conjunctive features during training. Iwakura and Okamoto (2008) proposed a boosting-based method that repeats the learning of rules represented by feature conjunctions. These methods, however, require us to tune the hyperparameter to trade model accuracy and the number of conjunctive features (memory footprint and training time); note that an accurate model may need many conjunctive features (in the hyponymy-relation extraction task, ℓ_1-LLM needed 15,828,122 features to obtain the best accuracy, 92.86%). Our method, on the other hand, takes all conjunctive features into consideration regardless of parameter N.

Dekel et al. (2008) and Cavallanti et al. (2007) improved the scalability of the (kernel) perceptron, by exploiting redundancy in the training data to bound the size of the support set to given threshold B ($\geq |\mathcal{S}_t|$). However, Orabona et al. (2009) reported that the models trained with these methods were just as accurate as a naive method that ceases training when $|\mathcal{S}_t|$ reaches the same threshold, B. They then proposed budget online learning based on PA-I, and it reduced the size of the support set to a tenth with a tolerable loss of accuracy. Their method, however, requires $\mathcal{O}(|\mathcal{S}_{t-1}|^2)$ time in updating the parameters in round t, which disables efficient training. We have proposed an orthogonal approach that exploits the data redundancy in evaluating the kernel to train the same model as the base learner.

6 Conclusion

In this paper, we proposed online learning with kernel slicing, aiming at resolving the space-time trade-off in training a classifier with many conjunctive features. The kernel slicing generalizes kernel splitting (Goldberg and Elhadad, 2008) in a purely feature-wise manner, to truly combine the merits of linear and kernel-based training. To improve the scalability of the training with redundant data in NLP, we reuse the temporal partial margins computed in past rounds and terminate unnecessary margin computations. Experiments on dependency parsing and hyponymy-relation extraction demonstrated that our method could train a classifier orders of magnitude faster than kernel-based learners, while retaining its space efficiency.

We will evaluate our method with ample labeled data obtained by the semi-supervised methods. The implementation of the proposed algorithm for kernel-based online learners is available from http://www.tkl.iis.u-tokyo.ac.jp/~ynaga/.

Acknowledgment We thank Susumu Yata for providing us practical lessons on the double-array trie, and thank Yoshimasa Tsuruoka for making his ℓ_1-LLM code available to us. We are also indebted to Nobuhiro Kaji and the anonymous reviewers for their valuable comments.

References

Ando, Rie Kubota and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.

Aoe, Jun'ichi. 1989. An efficient digital search algorithm by using a double-array structure. *IEEE Transactions on Software Engineering*, 15(9):1066–1077.

Bellare, Kedar, Partha Pratim Talukdar, Giridhar Kumaran, Fernando Pereira, Mark Liberman, Andrew McCallum, and Mark Dredze. 2007. Lightly-supervised attribute extraction. In *Proc. NIPS 2007 Workshop on Machine Learning for Web Search*.

Cavallanti, Giovanni, Nicolò Cesa-Bianchi, and Claudio Gentile. 2007. Tracking the best hyperplane with a simple budget perceptron. *Machine Learning*, 69(2-3):143–167.

Chang, Yin-Wen, Cho-Jui Hsieh, Kai-Wei Chang, Michael Ringgaard, and Chih-Jen Lin. 2010. Training and testing low-degree polynomial data mappings via linear SVM. *Journal of Machine Learning Research*, 11:1471–1490.

Crammer, Koby, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

Daumé III, Hal. 2006. *Practical Structured Learning Techniques for Natural Language Processing*. Ph.D. thesis, University of Southern California.

Daumé III, Hal. 2008. Cross-task knowledge-constrained self training. In *Proc. EMNLP 2008*, pages 680–688.

Dekel, Ofer, Shai Shalev-Shwartz, and Yoram Singer. 2008. The forgetron: A kernel-based perceptron on a budget. *SIAM Journal on Computing*, 37(5):1342–1372.

Freund, Yoav and Robert E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

Goldberg, Yoav and Michael Elhadad. 2008. splitSVM: fast, space-efficient, non-heuristic, polynomial kernel computation for NLP applications. In *Proc. ACL-08: HLT, Short Papers*, pages 237–240.

Isozaki, Hideki and Hideto Kazawa. 2002. Efficient support vector classifiers for named entity recognition. In *Proc. COLING 2002*, pages 1–7.

Iwakura, Tomoya and Seishi Okamoto. 2008. A fast boosting-based learner for feature-rich tagging and chunking. In *Proc. CoNLL 2008*, pages 17–24.

Kudo, Taku and Yuji Matsumoto. 2003. Fast methods for kernel-based text analysis. In *Proc. ACL 2003*, pages 24–31.

Liang, Percy, Hal Daumé III, and Dan Klein. 2008. Structure compilation: trading structure for features. In *Proc. ICML 2008*, pages 592–599.

Okanohara, Daisuke and Jun'ichi Tsujii. 2007. A discriminative language model with pseudo-negative samples. In *Proc. ACL 2007*, pages 73–80.

Okanohara, Daisuke and Jun'ichi Tsujii. 2009. Learning combination features with L_1 regularization. In *Proc. NAACL HLT 2009, Short Papers*, pages 97–100.

Orabona, Francesco, Joseph Keshet, and Barbara Caputo. 2009. Bounded kernel-based online learning. *Journal of Machine Learning Research*, 10:2643–2666.

Perkins, Simon, Kevin Lacker, and James Theiler. 2003. Grafting: fast, incremental feature selection by gradient descent in function space. *Journal of Machine Learning Research*, 3:1333–1356.

Sassano, Manabu. 2004. Linear-time dependency analysis for Japanese. In *Proc. COLING 2004*, pages 8–14.

Sumida, Asuka, Naoki Yoshinaga, and Kentaro Torisawa. 2008. Boosting precision and recall of hyponymy relation acquisition from hierarchical layouts in Wikipedia. In *Proc. LREC 2008*, pages 2462–2469.

Tsuruoka, Yoshimasa, Jun'ichi Tsujii, and Sophia Ananiadou. 2009. Stochastic gradient descent training for L1-regularized log-linear models with cumulative penalty. In *Proc. ACL-IJCNLP 2009*, pages 477–485.

Williams, Hugh E. and Justin Zobel. 1999. Compressing integers for fast file access. *The Computer Journal*, 42(3):193–201.

Wu, Yu-Chieh, Jie-Chi Yang, and Yue-Shi Lee. 2007. An approximate approach for training polynomial kernel SVMs in linear time. In *Proc. ACL 2007, Interactive Poster and Demonstration Sessions*, pages 65–68.

Yata, Susumu, Masahiro Tamura, Kazuhiro Morita, Masao Fuketa, and Jun'ichi Aoe. 2009. Sequential insertions and performance evaluations for double-arrays. In *Proc. the 71st National Convention of IPSJ*, pages 1263–1264. (In Japanese).

Yoshinaga, Naoki and Masaru Kitsuregawa. 2009. Polynomial to linear: efficient classification with conjunctive features. In *Proc. EMNLP 2009*, pages 1542–1551.

Discriminative Training for Near-Synonym Substitution

Liang-Chih Yu[1], Hsiu-Min Shih[2], Yu-Ling Lai[2], Jui-Feng Yeh[3] and Chung-Hsien Wu[4]
[1]Department of Information Management, Yuan Ze University
[2]Department of Mathematics, National Chung Cheng University
[3]Department of CSIE, National Chia-Yi University
[4]Department of CSIE, National Cheng Kung University
Contact: lcyu@saturn.yzu.edu.tw

Abstract

Near-synonyms are useful knowledge resources for many natural language applications such as query expansion for information retrieval (IR) and paraphrasing for text generation. However, near-synonyms are not necessarily interchangeable in contexts due to their specific usage and syntactic constraints. Accordingly, it is worth to develop algorithms to verify whether near-synonyms do match the given contexts. In this paper, we consider the near-synonym substitution task as a classification task, where a classifier is trained for each near-synonym set to classify test examples into one of the near-synonyms in the set. We also propose the use of discriminative training to improve classifiers by distinguishing positive and negative features for each near-synonym. Experimental results show that the proposed method achieves higher accuracy than both pointwise mutual information (PMI) and n-gram-based methods that have been used in previous studies.

1 Introduction

Near-synonym sets represent groups of words with similar meaning, which are useful knowledge resources for many natural language applications. For instance, they can be used for query expansion in information retrieval (IR) (Moldovan and Mihalcea, 2000; Bhogal et al., 2007), where a query term can be expanded by its near-synonyms to improve the recall rate. They can also be used in an intelligent thesaurus that can automatically suggest alternative words to avoid repeating the same word in the composing of text when there are suitable alternatives in its synonym set (Inkpen and Hirst, 2006; Inkpen, 2007). These near-synonym sets can be derived from manually constructed dictionaries such as WordNet (called synsets) (Fellbaum, 1998), EuroWordNet (Rodríguez et al., 1998), or clusters derived using statistical approaches (Lin, 1998).

Although the words in a near-synonym set have similar meaning, they are not necessarily interchangeable in practical use due to their specific usage and collocational constraints. Pearce (2001) presented an example of collocational constraints for the context " ____ coffee". In the given near-synonym set {strong, powerful}, the word "strong" is more suitable than "powerful" to fill the gap, since "powerful coffee" is an anti-collocation. Inkpen (2007) also presented several examples of collocations (e.g. ghastly mistake) and anti-collocations (e.g. ghastly error). Yu *et al*. (2007) described an example of the context mismatch problem for the context " ____ under the bay" and the near-synonym set {bridge, overpass, viaduct, tunnel} that represents the meaning of a physical structure that connects separate places by traversing an obstacle. The original word (target word) in the given context is "tunnel", and cannot be substituted by the other words in the same set since all the substitutions are semantically implausible. Accordingly, it is worth to develop algorithms to verify whether near-synonyms do match the given contexts. Applications can benefit from this ability to provide more effective services. For instance, a writing support system can assist users to select an alternative word that best fits a given context from a list of near-synonyms.

In measuring the substitutability of words, the co-occurrence information between a target word

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1254–1262,
Beijing, August 2010

(the gap) and its context words is commonly used in statistical approaches. Edmonds (1997) built a lexical co-occurrence network from 1989 Wall Street Journal to determine the near-synonym that is most typical or expected in a given context. Inkpen (2007) used the pointwise mutual information (PMI) formula to select the best near-synonym that can fill the gap in a given context. The PMI scores for each candidate near-synonym are computed using a larger web corpus, the Waterloo terabyte corpus, which can alleviate the data sparseness problem encountered in Edmonds' approach. Following Inkpen's approach, Gardiner and Dras (2007) also used the PMI formula with a different corpus (the Web 1T 5-gram corpus) to explore whether near-synonyms differ in attitude.

Yu *et al*. (2007) presented a method to compute the substitution scores for each near-synonym based on n-gram frequencies obtained by querying Google. A statistical test is then applied to determine whether or not a target word can be substituted by its near-synonyms. The dataset used in their experiments are derived from the OntoNotes copus (Hovy et al., 2006; Pradhan et al., 2007), where each near-synonym set corresponds to a *sense pool* in OntoNotes. Another direction to the task of near-synonym substitution is to identify the senses of a target word and its near-synonyms using word sense disambiguation (WSD), comparing whether they were of the same sense (McCarthy, 2002; Dagan et al., 2006). Dagan *et al.* (2006) described that the use of WSD is an indirect approach since it requires the intermediate sense identification step, and thus presented a sense matching technique to address the task directly.

In this paper, we consider the near-synonym substitution task as a classification task, where a classifier is trained for each near-synonym set to classify test examples into one of the near-synonyms in the set. However, near-synonyms share more common context words (features) than semantically dissimilar words in nature. Such similar contexts may decrease classifiers' ability to discriminate among near-synonyms. Therefore, we propose the use of a supervised discriminative training technique (Ohler et al., 1999; Kuo and Lee, 2003; Zhou and He, 2009) to improve classifiers by distinguishing positive and negative features for each near-synonym. To our best knowledge, this is the first study that uses discriminative training for near-synonym or lexical substitution. The basic idea of discriminative training herein is to adjust feature weights according to the minimum classification error (MCE) criterion. The features that contribute to discriminating among near-synonyms will receive a greater positive weight, whereas the noisy features will be penalized and might receive a negative weight. This re-weighting scheme helps increase the separation of the correct class against its competing classes, thus improves the classification performance.

Sentence: This will make the _____ message easier to interpret.

Original word: error

Near-synonym set: {error, mistake, oversight}

Figure 1. Example of a near-synonym set and a sentence to be evaluated.

The proposed supervised discriminative training is compared with two unsupervised methods, the PMI-based (Inkpen, 2007) and n-gram-based (Yu *et al.*, 2007) methods. The goal of the evaluation is described as follows. Given a near-synonym set and a sentence with one of the near-synonyms in it, the near-synonym is deleted to form a gap in this sentence. Figure 1 shows an example. Each method is then applied to predict an answer (best near-synonym) that can fill the gap. The possible candidates are all the near-synonyms (including the original word) in the given set. Ideally, the correct answers should be provided by human experts. However, such data is usually unavailable, especially for a large set of test examples. Therefore, we follow Inkpen's experiments to consider the original word as the correct answer. The proposed methods can then be evaluated by examining whether they can restore the original word by filling the gap with the best near-synonym.

The rest of this work is organized as follows. Section 2 describes the PMI and n-gram-based methods for near-synonym substitution. Section 3 presents the discriminative training technique. Section 4 summarizes comparative results. Conclusions are finally drawn in Section 5.

2 Unsupervised Methods

2.1 PMI-based method

The mutual information can measure the co-occurrence strength between a near-synonym and the words in a given context. A higher mutual information score indicates that the near-synonym fits well in the given context, thus is more likely to be the correct answer. The point-wise mutual information (Church and Hanks, 1991) between two words *x* and *y* is defined as

$$PMI(x,y) = \log_2 \frac{P(x,y)}{P(x)P(y)}, \quad (1)$$

where $P(x,y) = C(x,y)/N$ denotes the probability that *x* and *y* co-occur; $C(x,y)$ is the number of times *x* and *y* co-occur in the corpus, and *N* is the total number of words in the corpus. Similarly, $P(x) = C(x)/N$, where $C(x)$ is the number of times *x* occurs in the corpus, and $P(y) = C(y)/N$, where $C(y)$ is the number of times *y* occurs in the corpus. Therefore, (1) can be re-written as

$$PMI(x,y) = \log_2 \frac{C(x,y) \cdot N}{C(x) \cdot C(y)}. \quad (2)$$

Inkpen (2007) computed the PMI scores for each near-synonym using the Waterloo terabyte corpus and a context window of size 2*k* (*k*=2). Given a sentence *s* with a gap, $s = ...w_1...w_k \_\_\_\_ w_{k+1}...w_{2k}...$, the PMI score for a near-synonym NS_i to fill the gap is defined as

$$PMI(NS_j, s) = \sum_{i=1}^{k} PMI(NS_j, w_i) + \sum_{i=k+1}^{2k} PMI(NS_j, w_i). \quad (3)$$

The near-synonym with the highest score is considered as the answer. In this paper, we use the Web 1T 5-gram corpus to compute PMI scores, the same as in (Gardiner and Dras, 2007). The frequency counts $C(\cdot)$ are retrieved from this corpus in the same manner within the 5-gram boundary.

2.2 N-gram-based method

The n-grams can capture contiguous word associations in given contexts. Given a sentence with a gap, the n-gram scores for each near-synonym are computed as follows. First, all possible n-grams containing the gap are extracted from the sentence. Each near-synonym then fills the gap to compute a normalized frequency according to

$$Z(ngram^i_{NS_j}) = \frac{\log\left(C(ngram^i_{NS_j}) + 1\right)}{\log C(NS_j)}, \quad (4)$$

where $C(ngram^i_{NS_j})$ denotes the frequency of an n-gram containing a near-synonym, $C(NS_j)$ denotes the frequency of a near-synonym, and $Z(ngram^i_{NS_j})$ denotes the normalized frequency of an n-gram, which is used to reduce the effect of high frequencies on measuring n-gram scores. All of the above frequencies are retrieved from the Web 1T 5-gram corpus.

The n-gram score for a near-synonym to fill the gap is computed as

$$NGRAM(NS_j, s) = \frac{1}{R} \sum_{i=1}^{R} Z(ngram^i_{NS_j}), \quad (5)$$

where $NGRAM(NS_j, s)$ denotes the n-gram score of a near-synonym, which is computed by averaging the normalized frequencies of all the n-grams containing the near-synonym, and *R* is the total number of n-grams containing the near-synonym. Again, the near-synonym with the highest score is the proposed answer. We herein use the 4-gram frequencies to compute n-gram scores as Yu *et al.* (2007) reported that the use of 4-grams achieved higher performance than the use of bigrams and trigrams.

3 Discriminative Training

3.1 Classifier

The supervised classification technique can also be applied to for near-synonym substitution. Each near-synonym in a set corresponds to a class. The classifiers for each near-synonym set are built from the labeled training data, i.e., a collection of sentences containing the near-synonyms. Such training data is easy to obtain since it requires no human annotation. The training data used herein is collected by extracting the 5-grams containing the near-synonyms from the Web 1T 5-gram corpus. The features used for training are the words occurring in the context of the near-synonyms in the 5-grams.

	Example		
	1	2	3
$g_1(x,M)=\cos(x,m_1)$	0.9*	0.6*	0.8
$g_2(x,M)=\cos(x,m_2)$	0.3	0.5	0.3*
$g_3(x,M)=\cos(x,m_3)$	0.2	0.4	0.1
$\max_{j\neq k} g_i(x,M)=$	0.3	0.5	0.8
$d_k(x,M)=$	-0.6	-0.1	0.5
$l_k(x,M)=$ (γ=5)	0.047	0.378	0.924

Table 1. Examples of correct classification and misclassification. * denotes the scores of the correct class.

For each near-synonym set, an $F \times K$ feature-class matrix, denoted as *M*, is created for classification. The rows represent the *F* distinct words (features) and the columns represent the *K* near-synonyms (classes) in the same set. Each entry in the matrix, m_{ij}, represents a weight of word *i* respect to near-synonym *j*, which is computed as the number of times word *i* appears in the contexts of near-synonym *j* divided by the total number of context words of near-synonym *j*. This frequency-based weight can then be transformed into a probabilistic form, i.e., divided by the sum of the weights of word *i* respect to all near-synonyms. Each test sentence is also transformed into an *F*-dimensional feature vector. Let $x=[x_1,...,x_i,...,x_F]$ denote the feature vector of an input sentence. The classification is performed by computing the cosine similarity between *x* and the column vectors (near-synonyms) in the matrix, defined as

$$\begin{aligned} \hat{NS}_j &= \arg\max_j \cos(x,m_j) \\ &= \arg\max_j \frac{x \bullet m_j}{\|x\|\|m_j\|} \\ &= \arg\max_j \frac{\sum_{i=1}^{F} x_i m_{ij}}{\sqrt{\sum_{i=1}^{F} {x_i}^2}\sqrt{\sum_{i=1}^{F} {m_{ij}}^2}}, \end{aligned} \tag{6}$$

where m_j is the *j*-th column vector in the matrix *M*. The near-synonym with the highest cosine similarity score, $\hat{NS}_j$, is the predicted class of the classifier.

3.2 Minimum classification error criterion

According to the decision rule of the classifier, a classification error will occur when the near-synonym with the highest cosine score is not the correct class. Table 1 shows some examples, where Example 3 is an example of misclassification. On the other hand, although Example 2 is a correct classification, it might be an ambiguous case to classifiers since the scores are close among classes. Therefore, if we can increase the separation of the correct class from its competing ones, then the classification performance can be improved accordingly. This can be accomplished by adjusting the feature weights of the matrix *M* that have direct influence on the computation of cosine scores. The discriminative training performs the adjustment in the training phase according to the minimum classification error criterion. The detailed steps are as follows.

Given an input vector *x*, the classifier computes the cosine scores between *x* and each class using (6). The discriminant function for a class can thus be defined as the cosine measure; that is,

$$g_j(x,M)=\cos(x,m_j). \tag{7}$$

where *j* denotes a class in *K*. Since the correct class of each input vector is known in the training phase, we can determine whether or not the input vector is misclassified by comparing the discriminant function (cosine score) of the correct class against its competing classes. In the case of misclassification, the cosine score of the correct class will be smaller than the competing cosine scores. Let *k* be the correct class of *x*, the misclassification function can be defined as

$$d_k(x,M)=-g_k(x,M)+G_k(x,M), \tag{8}$$

where $g_k(x,M)$ is the discriminant function for the correct class *k*, and $G_k(x,M)$ is the anti-discriminant function that represents the other $K-1$ competing classes, defined as

$$G_k(x,M)=\left[\frac{1}{K-1}\sum_{j\neq k} g_j(x,M)^{\eta}\right]^{\frac{1}{\eta}}, \tag{9}$$

When $\eta=1$, the anti-discriminant function $G_k(x,M)$ is the average of all the competing cosine scores. With the increase of η, $G_k(x,M)$ is gradually dominated by the biggest

competing class. In the extreme case, i.e., $\eta \to \infty$, the anti-discriminant function becomes

$$G_k(x,M) = \max_{j \neq k} g_j(x,M). \quad (10)$$

The misclassification function in (8) can thus be rewritten as

$$d_k(x,M) = -g_k(x,M) + \max_{j \neq k} g_j(x,M), \quad (11)$$

In this case, the classification error is determined by comparing the discriminant function of the correct class against that of the biggest competing class. Obviously, $d_k(x,M) > 0$ implies a classification error. For instance, in Example 3, the discriminant function for the correct class is $g_2(x,M) = 0.3$, and that of the biggest competing class is $\max(g_1(x,M), g_3(x,M)) = 0.8$, thus the classification error is $d_k(x,M) = 0.5$. On the other hand, the classification error will be a negative value for correct classifications, as shown in Example 1 and 2.

Intuitively, a greater classification error also results in a greater loss. We herein use the sigmoid function as the class loss function; that is,

$$l_k(x,M) = l(d_k) = \frac{1}{1+\exp^{-\gamma d_k}}, \quad (12)$$

where γ is a constant that controls the slope of the sigmoid function. The sigmoid function maps the values of classification error within the range of 0 to 1. For correct classifications, a greater separation of the correct class from the biggest competing class leads to a greater negative value of d_k, i.e., a smaller classification error, resulting in a smaller loss tends asymptotically to 0 (Example 1), whereas a moderate loss is yielded if the separation was close (Example 2). For the cases of misclassification, a greater separation leads to a greater positive value of d_k, i.e., a greater classification error, resulting in a greater loss tends asymptotically to 1 (Example 3). The overall loss of the entire training set can then be estimated as

$$L(M) = \frac{1}{Q}\sum_{k=1}^{K}\sum_{x \in C_k} l_k(x,M), \quad (13)$$

where C_k denotes the set of training vectors of class k, and Q is the total number of vectors in the training set. The goal now is to minimize the loss. According to the above discussions on the three examples, to minimize the loss is to minimize the classification error, and to improve the separation of the correct class against its competing classes. This can be accomplished by adjusting the feature weights of the matrix M to distinguish positive and negative features for each class. We herein adopt a gradient descent method such as the generalized probabilistic descent (GPD) algorithm (Katagiri et al., 1998) to update the matrix M. The detailed steps are as follows.

Let the feature weights of the matrix M be the parameter set to be adjusted. The adjustment is performed iteratively according to the following update formula.

$$M^{(t+1)} = M^{(t)} - \varepsilon_t \nabla l_k(x^{(t)}, M^{(t)}), \quad (14)$$

where t denotes the t-th iteration, ε_t is an adjustment step of a small positive real number, and $\nabla l_k(x^{(t)}, M^{(t)})$ is the gradient of the loss function, which is computed by the following two parts

$$\nabla l_k(x^{(t)}, M^{(t)}) = \frac{\partial l_k}{\partial d_k}\frac{\partial d_k(x^{(t)}, M^{(t)})}{\partial m_{ij}}, \quad (15)$$

where

$$\frac{\partial l_k}{\partial d_k} = \gamma l_k(d_k)(1 - l_k(d_k)), \quad (16)$$

and from (7), (8), and (9),

$$\frac{\partial d_k(x^{(t)}, M^{(t)})}{\partial m_{ij}} = \begin{cases} -x_i, & \text{if } j = k \\ x_i \dfrac{G_k(x^{(t)}, M) g_j(x^{(t)}, M)^{\eta-1}}{\sum_{j \neq k} g_j(x^{(t)}, M)^{\eta}}, & \text{if } j \neq k \end{cases}, \quad (17)$$

where x_i is an element of the input vector x. By replacing $\nabla l_k(x_t, M_t)$ in (14) with the two parts in (15), at each iteration each feature weight m_{ij} in M is adjusted by

$$m_{ij}^{(t+1)} = \begin{cases} m_{ij}^{(t)} + \varepsilon_t \dfrac{\partial l_k}{\partial d_k} x_i, & \text{if } j = k \\ m_{ij}^{(t)} - \varepsilon_t \dfrac{\partial l_k}{\partial d_k} x_i \dfrac{G_k(x^{(t)}, M) g_j(x^{(t)}, M)^{\eta-1}}{\sum_{j \neq k} g_j(x^{(t)}, M)^{\eta}}, & \text{if } j \neq k \end{cases}. \quad (18)$$

The weight x_i represents whether or not a dimension word occurs in an input sentence. A zero

weight indicates that the dimension word does not occur in the input sentence, thus the corresponding dimension of each column vector will not be adjusted. On the contrary, the corresponding dimension of the column vector of the correct class ($j = k$) is adjusted by adding a value, while those of the competing classes ($j \neq k$) are adjusted by subtracting a value from them. After a sequence of adjustments over the training set, the positive and negative features can be distinguished by adjusting their weights that result in a greater positive or negative value for each of them. The separation of the correct class against its competing ones can thus be increased.

The weight adjustment in (18) is in proportion to the adjustment step ε_t and the slope of the sigmoid function $\partial l_k / \partial d_k$. The adjustment step ε_t can be determined empirically. As (16) shows, the slope $\partial l_k / \partial d_k$ converges asymptotically to zero as the classification error d_k approaches to a very large (or small) value. This leads to a small weight adjustment. For instance, the weight adjustment in Example 1 is small due to a small value of d_k, or, say, due to a large separation between the correct class and its competing ones. This is reasonable because classifiers often perform well in such cases. Similarly, the weight adjustment in Example 3 (misclassification) is also small due to a large value of d_k. A greater adjustment is not employed because such a large separation is difficult to be reduced significantly. Additionally, over-adjusting some features may introduce negative effects on other useful features in the matrix. Therefore, discriminative training is more effective on the cases with a moderate value of d_k, like Example 2. Such cases usually fall within the decision boundary and tend to be confusing to classifiers. Hence, improving the separation on such cases helps significantly improve the classification performance.

4 Experimental Results

4.1 Experiment setup

1) Data: The near-synonym sets used for experiments included the seven sets (Exp1) and the eleven sets (Exp2) used in the previous studies (Edmonds, 1997; Inkpen, 2007; Gardiner and Dras, 2007), as shown in Table 2. The Web 1T 5-gram corpus was used to build classifiers,

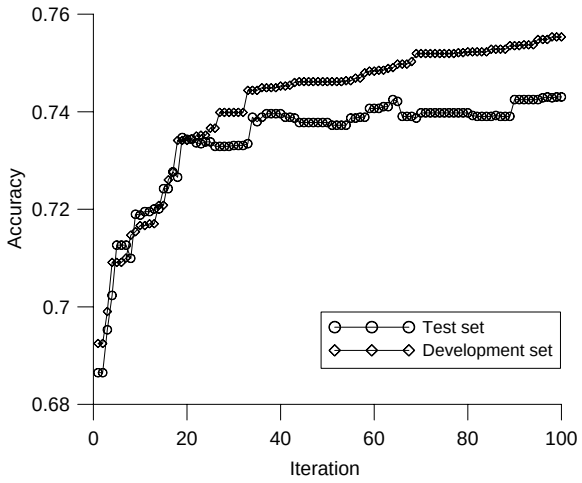


Figure 2. The change of classification accuracy during discriminative training.

where the corpus was randomly split into a training set, a development set, and a test set with an 8:1:1 ratio. For efficiency consideration, we randomly sampled up to 100 5-grams from the test set for each near-synonym. This sampling procedure was repeated five times for evaluation of the classifiers.

2) Classifiers: The classifiers were implemented using PMI, n-grams, and discriminative training (DT) methods, respectively.

PMI: Given a near-synonym set and a test 5-gram with a gap, the PMI scores for each near-synonym were calculated using (3), where the size of the context window *k* was set to 2. The frequency counts between each near-synonym and its context words were retrieved from the training set.

NGRAM: For each test 5-gram with a gap, all possible 4-grams containing the gap were first extracted (excluding punctuation marks). The averaged 4-gram scores for each near-synonym were then calculated using (5). Again, the frequency counts of the 4-grams were retrieved from the training set.

DT: For each near-synonym set, the matrix *M* was built from the training set. Each 5-gram in the development set was taken as input to iteratively compute the cosine score, loss, classification error, respectively, and finally to adjust the feature weights of *M*. The parameters of DT included η for the anti-discriminative function, γ

No.	Near-synonym set	No. of cases	Accuracy (%)			
			NGRAM	PMI	COS	DT
1.	difficult, hard, tough	300	58.60	61.67	60.13	63.13
2.	error, mistake, oversight	300	68.47	78.33	77.20	79.20
3.	job, task, duty	300	68.93	70.40	74.00	75.67
4.	responsibility, burden, obligation, commitment	400	69.80	66.95	68.75	69.55
5.	material, stuff, substance	300	70.20	67.93	71.07	75.13
6.	give, provide, offer	300	58.87	66.47	64.13	68.27
7.	settle, resolve	200	69.30	68.10	77.10	84.10
	Exp1	**2,100**	**66.33**	**68.50**	**69.94**	**72.89**
1.	benefit, advantage, favor, gain, profit	500	71.44	69.88	69.44	71.36
2.	low, gush, pour, run, spout, spurt, squirt, stream	800	65.45	65.00	68.68	71.08
3.	deficient, inadequate, poor, unsatisfactory	400	65.65	69.40	70.35	74.35
4.	afraid, aghast, alarmed, anxious, apprehensive, fearful, frightened, scared, terror-stricken*	789	49.84	44.74	47.00	49.33
5.	disapproval, animadversion*, aspersion*, blame, criticism, reprehension*	300	72.80	79.47	80.00	82.53
6.	mistake, blooper, blunder, boner, contretemps*, error, faux pas*, goof, slip, solecism*	618	62.27	59.61	68.41	71.65
7.	alcoholic, boozer*, drunk, drunkard, lush, sot	433	64.90	80.65	77.88	84.34
8.	leave, abandon, desert, forsake	400	65.85	66.05	69.35	74.35
9.	opponent, adversary, antagonist, competitor, enemy, foe, rival	700	58.51	59.51	63.31	67.14
10.	thin, lean, scrawny, skinny, slender, slim, spare, svelte, willowy*, wiry	734	57.74	61.99	55.72	64.58
11.	lie, falsehood, fib, prevarication*, rationalization, untruth	425	57.55	63.58	69.46	74.21
	Exp2	**6,099**	**61.69**	**63.32**	**65.15**	**69.26**

Table 2. Accuracy of classifiers on Exp1 (7 sets) and Exp2 (11 sets). The words marked with * are excluded from the experiments because their 5-grams are very rare in the corpus.

for the sigmoid function, and ε_t for the adjustment step. The settings, $\eta = 25$, $\gamma = 35$, and $\varepsilon_t = 10^{-3}$, were determined by performing DT for several iterations through the training set. These setting were used for the following experiments.

3) Evaluation metric: The answers proposed by each classifier are the near-synonyms with the highest score. The correct answers are the near-synonyms originally in the gap of the test 5-grams. The performance is measure by the accuracy, which is defined as the number of correct answers made by each classifier, divided by the total number of test 5-grams.

In the following sections, we first demonstrate the effect of DT on classification performance, followed by the comparison of the classifiers.

4.2 Evaluation on discriminative training

This experiment is to investigate the performance change during discriminative training. Figure 2 shows the accuracy at the first 100 iterations for both development set and test set, with the 8th set in Exp2 as an example. The accuracy increased rapidly in the first 20 iterations, and stabilized after the 40th iteration. The discriminative training is stopped until the accuracy has not been changed over 30 iterations or the 300th iteration has been reached.

Features	Near-synonym set		
	mistake	error	oversight
made	**0.076**	-0.004	-0.005
biggest	**0.074**	-0.001	-0.004
message	-0.004	**0.039**	-0.010
internal	0.001	**0.026**	-0.001
supervision	-0.001	-0.006	**0.031**
audit	-0.002	-0.003	**0.028**

Table 3. Example of feature weights after discriminative training.

Exp1	Rank 1	Rank 2	Diff.
NGRAM	66.33%	79.35%	+19.63%
PMI	68.50%	88.99%	+29.91%
COS	69.94%	89.93%	+28.58%
DT	72.89%	91.06%	+24.93%
Exp2	Rank 1	Rank 2	Diff.
NGRAM	61.69%	68.48%	+11.01%
PMI	63.32%	79.11%	+24.94%
COS	65.15%	80.52%	+23.59%
DT	69.26%	82.86%	+19.64%

Table 4. Accuracy of Rank 1 and Rank 2 for each classifier.

4.3 Comparative results

Table 2 shows the comparative results of the classification accuracy for the 18 near-synonym sets (Exp1 + Exp2). The accuracies for each near-synonym set were the average accuracies of the five randomly sampled test sets. The cosine measure without discrimination training (COS) was also considered for comparison. The results show that NGRAM performed worst among the four classifiers. The major reason is that not all 4-grams of the test examples can be found in the corpus. Instead of contiguous word associations used by NGRAM, PMI considers the words in the contexts independently to select the best synonyms. The results show that PMI achieved better performance than NGRAM. The two supervised methods, COS and DT, outperformed the two unsupervised methods, NGRAM and PMI. As indicated in the bold numbers, using the supervised method alone (without DT), COS yielded higher average accuracy by 5% and 2% over NGRAM and PMI, respectively, on Exp1, and by 6% and 3%, respectively, on Exp2. When DT was employed, the average accuracy was further improved by 4% and 6% on Exp1 and Exp2, respectively, compared with COS.

The use of DT can improve the classification accuracy mainly because it can adjust the feature weights iteratively to improve the separation between the correct class and its competing ones, which helps tackle the ambiguous test examples that fall within the decision boundary. Table 3 presents several positive and negative features for the near-synonym set {mistake, error, oversight}. The feature weights were adjusted according to their contributions to discriminating among the near-synonyms in the set. For instance, the features "made" and "biggest" both received a positive value for the class "mistake", and a negative value for the competing classes "error" and "oversight". These positive and negative weights help distinguish useful features from noisy ones for classifier training. On the other hand, if the feature weights were evenly distributed among the classes, these features would not be unlikely to contribute to the classification performance.

4.4 Accuracy of Rank 1 and Rank 2

The accuracy presented in Table 2 was computed based on the classification results at Rank 1, i.e., a test sample was considered correctly classified only if the near-synonym with the highest score was the word originally in the gap of the test sample. Similarly, the accuracy at Rank 2 can be computed by considering the top two near-synonyms proposed by each classifier. That is, if either the near-synonym with the highest or the second highest score was the correct answer, the test sample was considered correctly classified. Table 4 shows the accuracy of Rank 1 and Rank 2 for each classifier. The results show that the improvement of Rank 1 accuracy on Exp1 was about 20 to 30 percentage points, and was 25.76 in average. For Exp2, the average improvement was 19.80 percentage points

5 Conclusion

This work has presented the use of discriminative training for near-synonym substitution. The discriminative training can improve classification performance by iteratively re-weighting the positive and negative features for each class. This helps improve the separation of the correct class against its competing ones, making classifiers more effective on ambiguous cases close to the decision boundary. Experimental results show that the supervised discriminative training technique achieves higher accuracy than the two unsupervised methods, the PMI-based and n-gram-based methods. The availability of a large labeled training set also encourages the use of the proposed supervised method.

Future work will investigate on the use of multiple features for discriminating among near-synonyms. For instance, the predicate-argument structure, which can capture long-distance information, will be combined with currently used local contextual features to boost the classification performance. More experiments will also be conducted to evaluate classifiers' ability to rank multiple answers.

References

J. Bhogal, A. Macfarlane, and P. Smith. 2007. A Review of Ontology based Query Expansion. *Information Processing & Management*, 43(4):866-886.

K. Church and P. Hanks. 1991. Word Association Norms, Mutual Information and Lexicography. *Computational Linguistics*, 16(1):22-29.

I. Dagan, O. Glickman, A. Gliozzo, E. Marmorshtein, and C. Strapparava. 2006. Direct Word Sense Matching for Lexical Substitution. In *Proc. of COLING/ACL-06*, pages 449-456.

P. Edmonds. 1997. Choosing the Word Most Typical in Context Using a Lexical Co-occurrence Network. In *Proc. of ACL-97*, pages 507-509.

C. Fellbaum. 1998. WordNet: An Electronic Lexical Database. MIT Press, Cambridge, MA.

M. Gardiner and M. Dras. 2007. Exploring Approaches to Discriminating among Near-Synonyms, In *Proc. of the Australasian Technology Workshop*, pages 31-39.

E. H. Hovy, M. Marcus, M. Palmer, L. Ramshaw, and R. Weischedel. 2006. OntoNotes: The 90% Solution. In *Proc. of HLT/NAACL-06*, pages 57–60.

D. Inkpen. 2007. Near-Synonym Choice in an Intelligent Thesaurus. In *Proc. of NAACL/HLT-07*, pages 356-363.

D. Inkpen and G. Hirst. 2006. Building and Using a Lexical Knowledge-base of Near-Synonym Differences. *Computational Linguistics*, 32(2):1-39.

S. Katagiri, B. H. Juang, and C. H. Lee. 1998. Pattern Recognition Using a Family of Design Algorithms based upon the Generalized Probabilistic Descent Method, *Proc. of the IEEE*, 86(11):2345-2373.

H. K. J. Kuo and C. H. Lee. 2003. Discriminative Training of Natural Language Call Routers, *IEEE Trans. Speech and Audio Processing*, 11(1):24-35.

D. Lin. 1998. Automatic Retrieval and Clustering of Similar Words. In *Proc. of ACL/COLING-98*, pages 768-774.

D. McCarthy. 2002. Lexical Substitution as a Task for WSD Evaluation. In *Proc. of the SIGLEX/SENSEVAL Workshop on Word Sense Disambiguation at ACL-02*, pages 109-115.

D. Moldovan and R. Mihalcea. 2000. Using Wordnet and Lexical Operators to Improve Internet Searches. *IEEE Internet Computing*, 4(1):34-43.

U. Ohler, S. Harbeck, and H. Niemann. 1999. Discriminative Training of Language Model Classifiers, In *Proc. of Eurospeech-99*, pages 1607-1610.

D. Pearce. 2001. Synonymy in Collocation Extraction. In *Proc. of the Workshop on WordNet and Other Lexical Resources at NAACL-01*.

S. Pradhan, E. H. Hovy, M. Marcus, M. Palmer, L. Ramshaw, and R. Weischedel. 2007. OntoNotes: A Unified Relational Semantic Representation. In *Proc. of the First IEEE International Conference on Semantic Computing (ICSC-07)*, pages 517-524.

H. Rodríguez, S. Climent, P. Vossen, L. Bloksma, W. Peters, A. Alonge, F. Bertagna, and A. Roventint. 1998. The Top-Down Strategy for Building EeuroWordNet: Vocabulary Coverage, Base Concepts and Top Ontology, *Computers and the Humanities*, 32:117-159.

L. C. Yu, C. H. Wu, A. Philpot, and E. H. Hovy. 2007. OntoNotes: Sense Pool Verification Using Google N-gram and Statistical Tests, In *Proc. of the OntoLex Workshop at the 6th International Semantic Web Conference (ISWC-07)*.

D. Zhou and Y. He. 2009. Discriminative Training of the Hidden Vector State Model for Semantic Parsing, *IEEE Trans. Knowledge and Data Engineering*, 21(1):66-77.

Estimating Linear Models for Compositional Distributional Semantics

Fabio Massimo Zanzotto[1]
(1) Department of Computer Science
University of Rome "Tor Vergata"
zanzotto@info.uniroma2.it

Ioannis Korkontzelos
Department of Computer Science
University of York
johnkork@cs.york.ac.uk

Francesca Fallucchi[1,2]
(2) Università Telematica
"G. Marconi"
f.fallucchi@unimarconi.it

Suresh Manandhar
Department of Computer Science
University of York
suresh@cs.york.ac.uk

Abstract

In distributional semantics studies, there is a growing attention in compositionally determining the distributional meaning of word sequences. Yet, compositional distributional models depend on a large set of parameters that have not been explored. In this paper we propose a novel approach to estimate parameters for a class of compositional distributional models: the additive models. Our approach leverages on two main ideas. Firstly, a novel idea for extracting compositional distributional semantics examples. Secondly, an estimation method based on regression models for multiple dependent variables. Experiments demonstrate that our approach outperforms existing methods for determining a good model for compositional distributional semantics.

1 Introduction

Lexical distributional semantics has been largely used to model word meaning in many fields as computational linguistics (McCarthy and Carroll, 2003; Manning et al., 2008), linguistics (Harris, 1964), corpus linguistics (Firth, 1957), and cognitive research (Miller and Charles, 1991). The fundamental hypothesis is the distributional hypothesis (DH): "similar words share similar contexts" (Harris, 1964). Recently, this hypothesis has been operationally defined in many ways in the fields of physicology, computational linguistics, and information retrieval (Li et al., 2000; Pado and Lapata, 2007; Deerwester et al., 1990).

Given the successful application to words, distributional semantics has been extended to word sequences. This has happened in two ways: (1) via the reformulation of DH for specific word sequences (Lin and Pantel, 2001); and (2) via the definition of compositional distributional semantics (CDS) models (Mitchell and Lapata, 2008; Jones and Mewhort, 2007). These are two different ways of addressing the problem.

Lin and Pantel (2001) propose the *pattern distributional hypothesis* that extends the distributional hypothesis for specific patterns, i.e. word sequences representing partial verb phrases. Distributional meaning for these patterns is derived directly by looking to their occurrences in a corpus. Due to data sparsity, patterns of different length appear with very different frequencies in the corpus, affecting their statistics detrimentally. On the other hand, compositional distributional semantics (CDS) propose to obtain distributional meaning for sequences by composing the vectors of the words in the sequences (Mitchell and Lapata, 2008; Jones and Mewhort, 2007). This approach is fairly interesting as the distributional meaning of sequences of different length is obtained by composing distributional vectors of single words. Yet, many of these approaches have a large number of parameters that cannot be easily estimated.

In this paper we propose a novel approach to es-

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1263–1271,
Beijing, August 2010

timate parameters for additive compositional distributional semantics models. Our approach leverages on two main ideas. Firstly, a novel way for extracting compositional distributional semantics examples and counter-examples. Secondly, an estimation model that exploits these examples and determines an equation system that represents a regression problem with multiple dependent variables. We propose a method to estimate a solution of this equation system based on the Moore-Penrose pseudo-inverse matrices (Penrose, 1955).

The rest of the paper is organised as follows: Firstly, we shortly review existing compositional distributional semantics (CDS) models (Sec. 2). Then we describe our model for estimating CDS models parameters (Sec. 3). In succession, we introduce a way to extract compositional distributional semantics examples from dictionaries (Sec. 4). Then, we discuss the experimental set up and the results of our linear CDS model with estimated parameters with respect to existing CDS models (Sec. 5).

2 Models for compositional distributional semantics (CDS)

A CDS model is a function $\odot$ that computes the distributional vector of a sequence of words s by combining the distributional vectors of its component words $\mathsf{w}_1 \ldots \mathsf{w}_n$. Let $\odot(\mathsf{s})$ be the distributional vector describing s and $\vec{w}_i$ the distributional vectors describing its component word w_i. Then, the CDS model can be written as:

$$\odot(\mathsf{s}) = \odot(\mathsf{w}_1 \ldots \mathsf{w}_n) = \vec{w}_1 \odot \ldots \odot \vec{w}_n \quad (1)$$

This generic model has been fairly studied and many different functions have been proposed and tested.

Mitchell and Lapata (2008) propose the following general CDS model for 2-word sequences $\mathsf{s} = \mathsf{xy}$:

$$\odot(\mathsf{s}) = \odot(\mathsf{xy}) = f(\vec{x}, \vec{y}, R, K) \quad (2)$$

where $\vec{x}$ and $\vec{y}$ are respectively the distributional vectors of x and y, R is the particular syntactic and/or semantic relation connecting x and y, and, K represents the amount of background knowledge that the vector composition process takes into account. Two specialisations of the general CDS model are proposed: the *basic additive* model and the *basic multiplicative* model.

	vector dimensions				
	between	gap	process	social	two
contact	$< 11,$	$0,$	$3,$	$0,$	$11 >$
x: *close*	$< 27,$	$3,$	$2,$	$5,$	$24 >$
y: *interaction*	$< 23,$	$0,$	$3,$	$8,$	$4 >$

Table 1: Example of distributional frequency vectors for the triple $t = (\vec{contact}, \vec{close}, \vec{interaction})$

The *basic additive* model (BAM) is written as:

$$\odot(\mathsf{s}) = \alpha\vec{x} + \beta\vec{y} \quad (3)$$

where α and β are two scalar parameters. The simplistic parametrisation is $\alpha = \beta = 1$. For example, given the vectors $\vec{x}$ and $\vec{y}$ of Table 1, $\odot_{BAM}(\mathsf{s}) =< 50, 3, 5, 13, 28 >$.

The *basic multiplicative* model (BMM) is written as:

$$s_i = x_i y_i \quad (4)$$

where s_i, x_i, and y_i are the i-th dimensions of the vectors $\odot(\mathsf{s})$, $\vec{x}$, and $\vec{y}$, respectively. For the example of Table 1, $\odot_{BMM}(\mathsf{s}) =< 621, 0, 6, 40, 96 >$.

Erk and Padó (2008) look at the problem in a different way. Let the general distributional meaning of the word w be $\vec{w}$. Their model computes a different vector $\vec{w}_\mathsf{s}$ that represents the specific distributional meaning of w with respect to s, i.e.:

$$\vec{w}_\mathsf{s} = \oslash(\mathsf{w}, \mathsf{s}) \quad (5)$$

In general, this operator gives different vectors for each word w_i in the sequence s, i.e. $\oslash(\mathsf{w}_i, \mathsf{s}) \neq \oslash(\mathsf{w}_j, \mathsf{s})$ if $i \neq j$. It also gives different vectors for a word w_i appearing in different sequences s_k and s_l, i.e. $\oslash(\mathsf{w}_i, \mathsf{s}_k) \neq \oslash(\mathsf{w}_i, \mathsf{s}_l)$ if $k \neq l$.

The model of Erk and Padó (2008) was designed to *disambiguate* the distributional meaning of a word w in the context of the sequence s. However, substituting the word w with the semantic head h of s, allows to compute the distributional meaning of sequence s as shaped by the

word that is governing the sequence (c.f. Pollard and Sag (1994)). For example, the distributional meaning of the word sequence *eats mice* is governed by the verb *eats*. Following this model, the distributional vector $\odot(\mathsf{s})$ can be written as:

$$\odot(\mathsf{s}) \approx \oslash(\mathsf{h}, \mathsf{s}) \tag{6}$$

The function $\oslash(\mathsf{h}, \mathsf{s})$ explicitly uses the relation R and the knowledge K of the general equation 2, being based on the notion of selectional preferences. We exploit the model for sequences of two words s=xy where the two words are related with an oriented syntactic relation r (e.g. r=adj_modifier). For making the syntactic relation explicit, we indicate the sequence as: $\mathsf{s} = \mathsf{x} \overset{r}{\leftarrow} \mathsf{y}$.

Given a word w, the model has to keep track of its selectional preferences. Consequently, each word w is represented with a triple:

$$(\vec{w}, R_{\mathsf{w}}, R_{\mathsf{w}}^{-1}) \tag{7}$$

where $\vec{w}$ is the distributional vector of the word w, R_{w} is the set of the vectors representing the direct selectional preferences of the word w, and R_{w}^{-1} is the set of the vectors representing the indirect selectional preferences of the word w. Given a set of syntactic relations $\mathcal{R}$, the set R_{w} and R_{w}^{-1} contain respectively a selectional preference vector $R_{\mathsf{w}}(r)$ and $R_{\mathsf{w}}(r)^{-1}$ for each $r \in \mathcal{R}$. Selectional preferences are computed as in Erk (2007). If x is the semantic head of sequence s, then the model can be written as:

$$\odot(\mathsf{s}) = \oslash(\mathsf{x}, \mathsf{x} \overset{r}{\leftarrow} \mathsf{y}) = \vec{x} \odot R_{\mathsf{y}}(r) \tag{8}$$

Otherwise, if y is the semantic head:

$$\odot(\mathsf{s}) = \oslash(\mathsf{y}, \mathsf{x} \overset{r}{\leftarrow} \mathsf{y}) = \vec{y} \odot R_{\mathsf{x}}^{-1}(r) \tag{9}$$

$\odot$ is in both cases realised using BAM or BMM. We will call these models: *basic additive* model with selectional preferences (BAM-SP) and basic multiplicative model with selectional preferences (BMM-SP).

Both Mitchell and Lapata (2008) and Erk and Padó (2008) experimented with few empirically estimated parameters. Thus, the general additive CDS model has not been adequately explored.

3 Estimating Additive Compositional Semantics Models from Data

The generic *additive* model sums the vectors $\vec{x}$ and $\vec{y}$ in a new vector $\vec{z}$:

$$\odot(\mathsf{s}) = \vec{z} = A\vec{x} + B\vec{y} \tag{10}$$

where A and B are two square matrices capturing the relation R and the background knowledge K of equation 2. Writing matrices A and B by hand is impossible because of their large size. Estimating these matrices is neither a simple classification learning problem nor a simple regression problem. It is a regression problem with multiple dependent variables. In this section, we propose our model to solve this regression problem using a set of training examples E.

The set of training examples E contains triples of vectors $(\vec{z}, \vec{x}, \vec{y})$. $\vec{x}$ and $\vec{y}$ are the two distributional vectors of the words x and y. $\vec{z}$ is the expected distributional vector of the composition of $\vec{x}$ and $\vec{y}$. Note that for an ideal perfectly performing CDS model we can write $\vec{z} = \odot(\mathsf{xy})$. However, in general the expected vector $\vec{z}$ is not guaranteed to be equal to the composed one $\odot(\mathsf{xy})$. Figure 1 reports an example of these triples, i.e., $t = (\vec{contact}, \vec{close}, \vec{interaction})$, with the related distributional vectors. The construction of E is discussed in section 4.

In the rest of the section, we describe how the regression problem with multiple dependent variables can be solved with a linear equation system and we give a possible solution of this equation system. In the experimental section, we refer to our model as the estimated additive model (EAM).

3.1 Setting the linear equation system

The matrices A and B of equation 10 can be joined in a single matrix:

$$\vec{z} = \begin{pmatrix} A & B \end{pmatrix} \begin{pmatrix} \vec{x} \\ \vec{y} \end{pmatrix} \tag{11}$$

For the triple t of table 1, equation 11 is:

$$\vec{contact} = \begin{pmatrix} A & B \end{pmatrix} \begin{pmatrix} \vec{close} \\ \vec{interaction} \end{pmatrix} \tag{12}$$

and it can be rewritten as:

$$\begin{pmatrix} 11 \\ 0 \\ 3 \\ 0 \\ 11 \end{pmatrix} = \begin{pmatrix} A_{5\times5} & B_{5\times5} \end{pmatrix} \begin{pmatrix} 27 \\ 3 \\ 2 \\ 5 \\ 24 \\ 23 \\ 0 \\ 3 \\ 8 \\ 4 \end{pmatrix} \quad (13)$$

Focusing on matrix $\begin{pmatrix} AB \end{pmatrix}$, we can transpose the matrices as follows:

$$\begin{aligned} \vec{z}^T &= \left(\begin{pmatrix} A & B \end{pmatrix} \begin{pmatrix} \vec{x} \\ \vec{y} \end{pmatrix} \right)^T \\ &= \begin{pmatrix} \vec{x}^T & \vec{y}^T \end{pmatrix} \begin{pmatrix} A^T \\ B^T \end{pmatrix} \end{aligned} \quad (14)$$

Matrix $\begin{pmatrix} \vec{x}^T & \vec{y}^T \end{pmatrix}$ is known and matrix $\begin{pmatrix} A^T \\ B^T \end{pmatrix}$ is to be estimated.

Equation 14 is the prototype of our final equation system. The larger the matrix $\begin{pmatrix} AB \end{pmatrix}$ to be estimated, the more equations like 14 are needed. Given set E that contains n triples $(\vec{z}, \vec{x}, \vec{y})$, we can write the following system of equations:

$$\begin{pmatrix} \vec{z}_1^T \\ \vec{z}_2^T \\ \vdots \\ \vec{z}_n^T \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} \vec{x}_1^T & \vec{y}_1^T \end{pmatrix} \\ \begin{pmatrix} \vec{x}_2^T & \vec{y}_2^T \end{pmatrix} \\ \vdots \\ \begin{pmatrix} \vec{x}_n^T & \vec{y}_n^T \end{pmatrix} \end{pmatrix} \begin{pmatrix} A^T \\ B^T \end{pmatrix} \quad (15)$$

The vectors derived from the triples can be seen as two matrices of n rows, Z and $\begin{pmatrix} XY \end{pmatrix}$ related to $\vec{z}_i^T$ and $\begin{pmatrix} \vec{x}_i^T & \vec{y}_i^T \end{pmatrix}$, respectively. The overall equation system is then the following:

$$Z = \begin{pmatrix} X & Y \end{pmatrix} \begin{pmatrix} A^T \\ B^T \end{pmatrix} \quad (16)$$

This equation system represents the constraints that matrices A and B have to satisfy in order to be a possible linear CDS model that can at least describe seen examples. We will hereafter call $\Lambda = \begin{pmatrix} A & B \end{pmatrix}$ and $Q = \begin{pmatrix} X & Y \end{pmatrix}$. The system in equation 16 can be simplified as:

$$Z = Q\Lambda^T \quad (17)$$

As Q is a rectangular and singular matrix, it is not invertible and the system in equation 16 has no solutions. It is possible to use the principle of Least Square Estimation for computing an approximation solution. The idea is to compute the solution $\widehat{\Lambda}$ that minimises the residual norm, i.e.:

$$\widehat{\Lambda}^T = \arg\min_{\Lambda^T} \|Q\Lambda^T - Z\|^2 \quad (18)$$

One solution for this problem is the **Moore-Penrose pseudoinverse** Q^+ (Penrose, 1955) that gives the following final equation:

$$\widehat{\Lambda}^T = Q^+ Z \quad (19)$$

In the next section, we discuss how the **Moore-Penrose pseudoinverse** is obtained using singular value decomposition (SVD).

3.2 Computing the pseudo-inverse matrix

The pseudo-inverse matrix can provide an approximated solution even if the equation system has no solutions. We here compute the **Moore-Penrose pseudoinverse** using singular value decomposition (SVD) that is widely used in computational linguistics and information retrieval for reducing spaces (Deerwester et al., 1990).

Moore-Penrose pseudoinverse (Penrose, 1955) is computed in the following way. Let the original matrix Q have n rows and m columns and be of rank r. The SVD decomposition of the original matrix Q is $Q = U\Sigma V^T$ where Σ is a square diagonal matrix of dimension r. Then, the pseudo-inverse matrix that minimises the equation 18 is:

$$Q^+ = V\Sigma^+ U^T \quad (20)$$

where the diagonal matrix Σ^+ is the $r \times r$ transposed matrix of Σ having as diagonal elements the reciprocals of the singular values $\frac{1}{\delta_1}, \frac{1}{\delta_2}, ..., \frac{1}{\delta_r}$ of Σ.

Using SVD to compute the pseudo-inverse matrix allows for different approximations (Fallucchi and Zanzotto, 2009). The algorithm for computing the singular value decomposition is iterative (Golub and Kahan, 1965). Firstly derived dimensions have higher singular value. Then, dimension k is more informative than dimension $k' > k$. We can consider different values for k to obtain different SVD for the approximations Q_k^+ of the original matrix Q^+ in equation 20), i.e.:

$$Q_k^+ = V_{n\times k}\Sigma_{k\times k}^+ U_{k\times m}^T \quad (21)$$

where Q_k^+ is a matrix n by m obtained considering the first k singular values.

4 Building positive and negative examples

As explained in the previous section, estimating CDS models, needs a set of triples E, similar to triple t of table 1. This set E should contain positive examples in the form of triples $(\vec{z}_i, \vec{x}_i, \vec{y}_i)$. Examples are positive in the sense that $\vec{z}_i = \odot(\mathsf{xy})$ for an ideal CDS. There are no available sets to contain such triples, with the exception of the set used in Mitchell and Lapata (2008) which is designed only for testing purposes. It contains similar and dissimilar pairs of sequences (s_1,s_2) where each sequence is a verb-noun pair (v_i,n_i). From the positive part of this set, we can only derive quadruples where $\odot(\mathsf{v}_1\mathsf{n}_1) \approx \odot(\mathsf{v}_2\mathsf{n}_2)$ but we cannot derive the ideal resulting vector of the composition $\odot(\mathsf{v}_i\mathsf{n}_i)$. Sets used to test multiword expression (MWE) detection models (e.g., (Schone and Jurafsky, 2001; Nicholson and Baldwin, 2008; Kim and Baldwin, 2008; Cook et al., 2008; Villavicencio, 2003; Korkontzelos and Manandhar, 2009)) are again not useful as containing only valid MWE that cannot be used to determine the set of training triples needed here.

As a result, we need a novel idea to build sets of triples to train CDS models. We can leverage on knowledge stored in dictionaries. In the rest of the section, we describe how we build the positive example set E and a control negative example set NE. Elements of the two sets are pairs (t,s) where t is a target word s is a sequence of words. t is the word that represent the distributional meaning of s in the case of E. Contrarily, t is totally unrelated to the distributional meaning of s in NE. The sets E and NE can be used both for training and for testing. In the testing phase, we can use these sets to determine whether a CDS model is good or not and to compare different CDS models.

4.1 Building Positive Examples using Dictionaries

Dictionaries as natural repositories of equivalent expressions can be used to extract positive examples for training and testing CDS models. The basic idea is the following: dictionary entries are *declarations* of equivalence. Words or, occasionally, multi-word expressions t are declared to be semantically similar to their definition sequences s. This happens at least for some sense of the defined words. We can then observe that $\mathsf{t} \approx \mathsf{s}$. For example, we report some sample definitions of *contact* and *high life*:

target word (t)	definition sequence (s)
contact	close interaction
high life	excessive spending

In the first case, a word, i.e. *contact*, is semantically similar to a two-word expression, i.e. *close interaction*. In the second case, two two-word expressions are semantically similar.

Then, the pairs (t, s) can be used to model positive cases of compositional distributional semantics as we know that the word sequence s is compositional and it describes the meaning of the word t. The distributional meaning $\vec{t}$ of t is the expected distributional meaning of s. Consequently, the vector $\vec{t}$ is what the CDS model $\odot(\mathsf{s})$ should compositionally obtain from the vectors of the components $\vec{s_1} \ldots \vec{s_m}$ of s. This way of extracting similar expressions has some interesting properties:

First property Defined words t are generally single words. Thus, we can extract stable and meaningful distributional vectors for these words and then compare them to the distributional vectors composed by CDS model. This is an important property as we cannot compare directly the distributional vector $\vec{s}$ of a word sequence s and the vector $\odot(\mathsf{s})$ obtained by composing its components. As the word sequence s grows in length, the reliability of the vector $\vec{s}$ decreases since the sequence s becomes rarer.

Second property Definitions s have a large variety of different syntactic structures ranging from simple structures as Adjective-Noun to more complex ones. This gives the possibility to train and test CDS models that take into account syntax. Table 2 represents the distribution of the more frequent syntactic structures in the definitions of WordNet[1] (Miller, 1995).

[1]Definitions were extracted from WordNet 3.0 and were parsed with the Charniak parser (Charniak, 2000)

Freq.	*Structure*
2635	(FRAG (PP (IN) (NP (DT) (JJ) (NN))))
833	(NP (DT) (JJ) (NN))
811	(NP (NNS))
645	(NP (NNP))
623	(S (VP (VB) (ADVP (RB))))
610	(NP (JJ) (NN))
595	(NP (NP (DT) (NN)) (PP (IN) (NP (NN))))
478	(NP (NP (DT) (NN)) (PP (IN) (NP (NNP))))
451	(FRAG (PP (IN) (NP (NN))))
419	(FRAG (RB) (ADJP (JJ)))
375	(S (VP (VB) (PP (IN) (NP (DT) (NN)))))
363	(S (VP (VB) (PP (IN) (NP (NN)))))
342	(NP (NP (DT) (NN)) (PP (IN) (NP (DT) (NN))))
341	(NP (DT) (JJ) (JJ) (NN))
330	(ADJP (RB) (JJ))
307	(NP (JJ) (NNS))
244	(NP (DT) (NN) (NN))
241	(S (NP (NN)) (NP (NP (NNS)) (PP (IN) (NP (DT) (NNP)))))
239	(NP (NP (DT) (JJ) (NN)) (PP (IN) (NP (DT) (NN))))

Table 2: Top 20 syntactic structures of WordNet definitions

4.2 Extracting Negative Examples from Word Etymology

In order to devise complete training and testing sets for CDS models, we need to find a sensible way to extract negative examples. An option is to randomly generate totally unrelated triples for the negative examples set, NE. In this case, due to data sparseness NE would mostly contain triples $(\vec{z}, \vec{x}, \vec{y})$ where it is expected that $\vec{z} \neq \odot(\mathsf{xy})$. Yet, these can be too generic and too loosely related to be interesting cases.

Instead we attempt to extract sets of negative pairs (t,s) comparable with the one used for building the training set E. The target word t should be a single word and s should be a sequence of words. The latter should be a sequence of words related by construction to t but the meaning of t and s should be unrelated.

The idea is the following: many words are etymologically derived from very old or ancient words. These words represent a collocation which is in general not related to the meaning of the target word. For example, the word *philosophy* derives from two Greek words *philos* (beloved) and *sophia* (wisdom). However, the use of the word *philosophy* in not related to the collocation *beloved wisdom*. This word has lost its original compositional meaning. The following table shows some more etymologically complex words along with the compositionally unrelated collocations:

target word	compositionally unrelated seq.
municipal	receive duty
octopus	eight foot

As the examples suggest, we are able to build a set NE with features similar to the features of N. In particular, each target word is paired with a related word sequence derived from its etymology. These etymologically complex words are unrelated to the corresponding compositional collocations. To derive a set NE with the above characteristics we can use dictionaries containing etymological information as Wiktionary[2].

5 Experimental evaluation

In the previous sections, we presented the estimated additive model (EAM): our approach to estimate the parameters of a generic additive model for CDS. In this section, we experiment with this model to determine whether it performs better than existing models: the basic additive model (BAM), the basic multiplicative model (BMM), the basic additive model with selectional preferences (BAM-SP), and the basic multiplicative model with selectional preferences (BMM-SP) (c.f. Sec. 2). In succession, we explore whether our estimated additive model (EAM) is better than any possible BAM obtained with parameter adjustment. In the rest of the section, we firstly give the experimental setup and then we discuss the experiments and the results.

5.1 Experimental setup

Our experiments aim to compare compositional distributional semantic (CDS) models $\odot$ with respect to their ability of detecting statistically significant difference between sets E and NE. In particular, the average similarity $sim(\vec{z}, \odot(\mathsf{xy}))$ for $(\vec{z}, \vec{x}, \vec{y}) \in E$ should be significantly different from $sim(\vec{z}, \odot(\mathsf{xy}))$ for $(\vec{z}, \vec{x}, \vec{y}) \in NE$. In this section, we describe the chosen similarity measure sim, statistical significance testing and construction details for the training and testing set.

Cosine similarity was used to compare the context vector $\vec{z}$ representing the target word z with the composed vector $\odot(\mathsf{xy})$ representing the context vector of sequence x y. Cosine similarity be-

[2] http://www.wiktionary.org

tween two vectors $\vec{x}$ and $\vec{y}$ of the same dimension is defined as:

$$sim(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\| \, \|\vec{y}\|} \quad (22)$$

where $\cdot$ is the dot product and $\|\vec{a}\|$ is the magnitude of vector $\vec{a}$ computed the Euclidean norm.

To evaluate whether a CDS model distinguishes positive examples E from negative examples NE, we test if the distribution of similarities $sim(\vec{z}, \odot(\mathsf{xy}))$ for $(\vec{z}, \vec{x}, \vec{y}) \in E$ is statistically different from the distribution of the same similarities for $(\vec{z}, \vec{x}, \vec{y}) \in NE$. For this purpose, we used Student's t-test for two independent samples of different sizes. t-test assumes that the two distributions are Gaussian and determines the probability that they are similar, i.e., derive from the same underlying distribution. Low probabilities indicate that the distributions are highly dissimilar and that the corresponding CDS model performs well, as it detects statistically different similarities for the positive set E and the negative set NE.

Based on the null hypothesis that the means of the two samples are equal, $\mu_1 = \mu_2$, Student's t-test takes into account the sizes N, means M and variances s^2 of the two samples to compute the following value:

$$t = (M_1 - M_2) \sqrt[-1]{\frac{2(s_1^2 + s_2^2)}{df * N_h}} \quad (23)$$

where $df = N_1 + N_2 - 2$ stands for the degrees of freedom and $N_h = 2(N_1^{-1} + N_2^{-1})^{-1}$ is the harmonic mean of the sample sizes. Given the statistic t and the degrees of freedom df, we can compute the corresponding p-value, i.e., the probability that the two samples derive from the same distribution. The null hypothesis can be rejected if the p-value is below the chosen threshold of statistical significance (usually 0.1, 0.05 or 0.01), otherwise it is accepted. In our case, rejecting the null hypothesis means that the similarity values of instances of E are significantly different from instances of NE, and that the corresponding CDS model perform well. p-value can be used as a performance ranking function for CDS models.

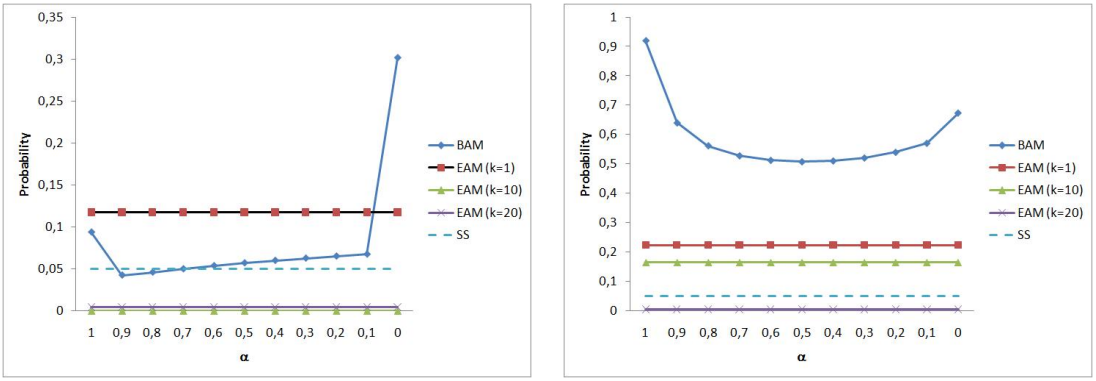
We constructed two sets of instances: (a) a set containing Adjective-Noun or Noun-Noun sequences (*NN* set); and (b) a set containing Verb-Noun sequences (*VN* set). Capturing different syntactic relations, these two sets can support that our results are independent from the syntactic relation between the words of each sequence. For each set, we used WordNet for extracting positive examples E and Wiktionary for extracting negative examples NE as described in Section 4. We obtained the following sets: (a) *NN* consists of 1065 word-sequence pairs from WordNet definitions and 377 pairs extracted from Wiktionary; and (b) *VN* consists of 161 word-sequence pairs from WordNet definitions and 111 pairs extracted from Wiktionary. We have then divided these two sets in two parts of 50% each, for training and testing. Instances of the training part of E have been used to estimate matrices A and B for model EAM, while the testing parts have been used for testing all models. Frequency vectors for all single words occurring in the above pairs were constructed from the British National Corpus using sentences as contextual windows and words as features. The resulting space has 689191 features.

	NN set	*VN* set
BAM	0.05690	0.50753
BMM	0.20262	0.37523
BAM-SP	0.42574	0.01710
BMM-SP	<1.00E-10	0.23552
EAM (k=20)	0.00431	0.00453

Table 3: Probability of confusing E and NE with different CDS models

5.2 Results and Analysis

The first set of experiments compares EAM with other existing CDS models: BAM, BMM, BAM-SP, and BMM-SP. Results are shown in Table 3. The table reports the p-value, i.e., the probability of confusing the positive set E and the negative set NE for all models. Lower probabilities characterise better models. Probabilities below 0.05 indicate that the model detects a statistically significant difference between sets E and NE. EAM has been computed with $k = 20$ different dimensions for the pseudo-inverse matrix. The two basic additive models (BAM and BAM-SP) have been computed for $\alpha = \beta = 1$.



NN set *VN set*

Figure 1: p-values of BAM with different values for parameter α (where $\beta = 1 - \alpha$) and of EAM for different approximations of the SVD pseudo-inverse matrix (k)

The first observation is that EAM models significantly separate positive from negative examples for both sets. This is not the case for any of the other models. Only, the selectional preferences based models in two cases have this property, but this cannot be generalised: BAM-SP on the *VN* set and BMM-SP on the *NN* set. In general, these models do not offer the possibility of separating positive from negative examples.

In the second set of experiments, we attempt to investigate whether simple parameter adjustment of BAM can perform better than EAM. Results are shown in figure 1. Plots show the basic additive model (BAM) with different values for parameter α (where $\beta = 1 - \alpha$) and EAM computed for different approximations of the SVD pseudo-inverse matrix (i.e., with different k). The x-axis of the plots represents parameter α and the y-axis represents the probability of confusing the positive set E and the negative set NE. The representation focuses on the performance of BAM with respect to different α values. The performance of EAM for different k values is represented with horizontal lines. Probabilities of different models are directly comparable. Line SS represents the threshold of statistical significance; the value below which the detected difference between the E and NE sets becomes statistically significant.

Experimental results show some interesting facts: While BAM for $\alpha > 0$ perform better than EAM computed with $k = 1$ in the NN set, they do not perform better in the VN set. EAM with $k = 1$ has 1 degree of freedom corresponding to 1 parameter, the same as BAM. The parameter of EAM is tuned on the training set, in contrast to α, the parameter of BAM. Increasing the number of considered dimensions, k of EAM, estimated models outperform BAM for all values of parameter α. Moreover, EAM detect a statistically significant difference between the E and the NE sets for $k \geq 10$ and $k = 20$ for the *NN set* and the *VN set* set, respectively. Simple parametrisation of a BAM does not outperform the proposed estimated additive model.

6 Conclusions

In this paper, we presented an innovative method to estimate linear compositional distributional semantics models. The core of our approach consists on two parts: (1) providing a method to estimate the regression problem with multiple dependent variables and (2) providing a training set derived from dictionary definitions. Experiments showed that our model is highly competitive with respect to state-of-the-art models for compositional distributional semantics.

References

Charniak, Eugene. 2000. A maximum-entropy-inspired parser. In *proceedings of the 1st NAACL*, pages 132–139, Seattle, Washington.

Cook, Paul, Afsaneh Fazly, and Suzanne Stevenson. 2008. The VNC-Tokens Dataset. In *proceedings of the LREC Workshop: Towards a Shared Task for Multiword Expressions (MWE 2008)*, Marrakech, Morocco.

Deerwester, Scott C., Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407.

Erk, Katrin and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 897–906. Association for Computational Linguistics.

Erk, Katrin. 2007. A simple, similarity-based model for selectional preferences. In *proceedings of ACL*. Association for Computer Linguistics.

Fallucchi, Francesca and Fabio Massimo Zanzotto. 2009. SVD feature selection for probabilistic taxonomy learning. In *proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, pages 66–73. Association for Computational Linguistics, Athens, Greece.

Firth, John R. 1957. *Papers in Linguistics.* Oxford University Press, London.

Golub, Gene and William Kahan. 1965. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis*, 2(2):205–224.

Harris, Zellig. 1964. Distributional structure. In Katz, Jerrold J. and Jerry A. Fodor, editors, *The Philosophy of Linguistics*, New York. Oxford University Press.

Jones, Michael N. and Douglas J. K. Mewhort. 2007. Representing word meaning and order information in a composite holographic lexicon. *Psychological Review*, 114:1–37.

Kim, Su N. and Timothy Baldwin. 2008. Standardised evaluation of english noun compound interpretation. In *proceedings of the LREC Workshop: Towards a Shared Task for Multiword Expressions (MWE 2008)*, pages 39–42, Marrakech, Morocco.

Korkontzelos, Ioannis and Suresh Manandhar. 2009. Detecting compositionality in multi-word expressions. In *proceedings of ACL-IJCNLP 2009*, Singapore.

Li, Ping, Curt Burgess, and Kevin Lund. 2000. The acquisition of word meaning through global lexical co-occurrences. In *proceedings of the 31st Child Language Research Forum*.

Lin, Dekang and Patrick Pantel. 2001. DIRT-discovery of inference rules from text. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD-01)*. San Francisco, CA.

Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK.

McCarthy, Diana and John Carroll. 2003. Disambiguating nouns, verbs, and adjectives using automatically acquired selectional preferences. *Computational Linguistics*, 29(4):639–654.

Miller, George A. and Walter G. Charles. 1991. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, VI:1–28.

Miller, George A. 1995. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41.

Mitchell, Jeff and Mirella Lapata. 2008. Vector-based models of semantic composition. In *proceedings of ACL-08: HLT*, pages 236–244, Columbus, Ohio. Association for Computational Linguistics.

Nicholson, Jeremy and Timothy Baldwin. 2008. Interpreting compound nominalisations. In *proceedings of the LREC Workshop: Towards a Shared Task for Multiword Expressions (MWE 2008)*, pages 43–45, Marrakech, Morocco.

Pado, Sebastian and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

Penrose, Roger. 1955. A generalized inverse for matrices. In *Proceedings of Cambridge Philosophical Society*.

Pollard, Carl J. and Ivan A. Sag. 1994. *Head-driven Phrase Structured Grammar*. Chicago CSLI, Stanford.

Schone, Patrick and Daniel Jurafsky. 2001. Is knowledge-free induction of multiword unit dictionary headwords a solved problem? In Lee, Lillian and Donna Harman, editors, *proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 100–108.

Villavicencio, Aline. 2003. Verb-particle constructions and lexical resources. In *proceedings of the ACL 2003 workshop on Multiword expressions*, pages 57–64, Morristown, NJ, USA. Association for Computational Linguistics.

Grouping Product Features Using Semi-Supervised Learning with Soft-Constraints*

Zhongwu Zhai[†], **Bing Liu**[‡], **Hua Xu**[†] **and Peifa Jia**[†]

[†]State Key Lab of Intelligent Tech. & Sys.
Tsinghua National Lab for Info. Sci. and Tech.
Dept. of Comp. Sci. & Tech., Tsinghua Univ.
zhaizhongwu@gmail.com

[‡]Dept. of Comp. Sci.
University of Illinois at Chicago
liub@cs.uic.edu

Abstract

In opinion mining of product reviews, one often wants to produce a summary of opinions based on product features/attributes. However, for the same feature, people can express it with different words and phrases. To produce a meaningful summary, these words and phrases, which are domain synonyms, need to be grouped under the same feature group. This paper proposes a constrained semi-supervised learning method to solve the problem. Experimental results using reviews from five different domains show that the proposed method is competent for the task. It outperforms the original EM and the state-of-the-art existing methods by a large margin.

1 Introduction

One form of opinion mining in product reviews is to produce a feature-based summary (Hu and Liu, 2004a; Liu, 2010). In this model, product features are first identified, and positive and negative opinions on them are aggregated to produce a summary on the features. Features of a product are attributes, components and other aspects of the product, e.g., "picture quality", "battery life" and "zoom" of a digital camera.

In reviews (or any writings), people often use different words and phrases to describe the same product feature. For example, "picture" and "photo" refer to the same feature for cameras. Grouping such synonyms is critical for effective opinion summary. Although WorldNet and other thesaurus dictionaries can help to some extent, they are far from sufficient due to a few reasons. First, many words and phrases that are not synonyms in a dictionary may refer to the same feature in an application domain. For example, "appearance" and "design" are not synonymous, but they can indicate the same feature, *design*. Second, many synonyms are domain dependent. For example, "movie" and "picture" are synonyms in movie reviews, but they are not synonyms in camera reviews as "picture" is more likely to be synonymous to "photo" while "movie" to "video". Third, determining which expressions indicate the same feature can be dependent on the user's application need. For example, in car reviews, internal design and external design can be regarded as two separate features, but can also be regarded as one feature, called "design", based to the level of details that the user needs to study. In camera reviews, one may want to study battery as a whole (one feature), or as more than one feature, e.g., battery weight, and battery life. Due to this reason, in applications the user needs to be involved in synonym grouping.

Before going further, let us introduce two concepts, *feature group* and *feature expression*. Feature group (or *feature* for short) is the name of a feature (given by the user), while a feature expression of a feature is a word or phrase that actually appears in a review to indicate the feature. For example, a feature group could be named "picture quality", but there are many possible expressions indicating the feature, e.g., "picture", "photo", "image", and even the "picture quality" itself. All the feature expressions in a feature group signify the same feature.

Grouping feature expressions manually into suitable groups is time consuming as there are

*Supported by National Natural Science Foundation of China (Grant No: 60875073).

This work was done when the first author was visiting Bing Liu's group at the University of Illinois at Chicago.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1272–1280,
Beijing, August 2010

often hundreds of feature expressions. This paper helps the user to perform the task more efficiently. To focus our research, we assume that feature expressions have been discovered from a review corpus by an existing system such as those in (Hu and Liu, 2004b; Popescu and Etzioni, 2005; Kim and Hovy, 2006; Kobayashi *et al.*, 2007; Mei *et al.*, 2007; Stoyanov and Cardie, 2008; Jin *et al.*, 2009; Ku *et al.*, 2009).

To reflect the user needs, he/she can manually label a small number of seeds for each feature group. The feature groups are also provided by the user based on his/her application needs. The system then assigns the rest of the feature expressions to suitable groups. To the best of our knowledge, this problem has not been studied in opinion mining (Pang and Lee, 2008).

The problem can be formulated as semi-supervised learning. The small set of seeds labeled by the user is the labeled data, and the rest of the discovered feature expressions are the unlabeled data. This is the transductive setting (Joachims, 1999) because the unlabeled set is used in learning and also in testing since our objective is to assign unlabeled expressions to the right feature groups.

Any semi-supervised learning method can be applied to tackle the problem. In this work, we use the Expectation-Maximization (EM) algorithm (Dempster *et al.*, 1977). Specifically, we use the naïve Bayesian EM formulation in (Nigam *et al.*, 2000), which runs a Bayesian classifier iteratively on the labeled and unlabeled data until the probabilities for the unlabeled data converge. When the algorithm ends, each unlabeled example is assigned a posterior probability of belonging to each group.

However, we can do better since the EM algorithm only achieves local optimal. What local optimal it achieves depends on the initialization, i.e., the initial seeds. We show that some prior knowledge can help provide a better initialization, and consequently generate better grouping results. Thus, we propose to create another set of data extracted from the unlabeled set based on two pieces of natural language knowledge:

1. Feature expressions sharing some common words are likely to belong to the same group, e.g., "battery life" and "battery power".
2. Feature expressions that are synonyms in a dictionary are likely to belong to the same group, e.g., "movie" and "picture".

We call these two pieces of prior knowledge *soft constraints* because they constrain the feature expressions to be in the same feature group. The constraints are soft (rather than hard) as they can be relaxed in the learning process. This relaxation is important because the above two constraints can result in wrong groupings. The EM algorithm is allowed to re-assign them to other groups in the learning process.

We call the proposed framework constrained semi-supervised learning. Since we use EM and soft constraints, we call the proposed method *SC-EM*. Clearly, the problem can also be attempted using some other techniques, e.g., topic modeling (e.g, LDA (Blei *et al.*, 2003)), or clustering using distributional similarity (Pereira *et al.*, 1993; Lin, 1998; Chen *et al.*, 2006; Sahami and Heilman, 2006). However, our results show that these methods do not perform as well.

The input to the proposed algorithm consists of: a set of reviews R, and a set of discovered feature expressions F from R (using an existing algorithm). The user labels a small set of feature expressions, i.e., assigning them to the user-specified feature groups. The system then assigns the rest of the discovered features to the feature groups. EM is run using the distributional (or surrounding words) contexts of feature expressions in review set R to build a naïve Bayesian classifier in each iteration.

Our evaluation was conducted using reviews from 5 different domains (insurance, mattress, vacuum, car and home-theater). The results show that the proposed method outperforms different variations of the topic modeling method LDA, *k*-means clustering, and the recent unsupervised feature grouping method mLSA.

In summary, this paper makes three main contributions:

1. It proposes a new sub-problem of opinion mining, i.e., grouping feature expressions in the context of semi-supervised learning. Although there are existing methods for solving the problem based on unsupervised learning, we argue that for practical use some form of supervision from the user is necessary to let the system know what the user wants.
2. An EM formulation is used to solve the problem. We augment EM with two soft constraints. These constraints help guide EM to

produce better solutions. We note that these constraints can be relaxed in the process to correct the imperfection of the constraints.

3. It is shown experimentally the new method outperforms the main existing state-of-the-art methods that can be applied to the task.

2 Related Work

This work is mainly related to existing research on synonyms grouping, which clusters words and phrases based on some form of similarity.

The methods for measuring word similarity can be classified into two main types (Agirre *et al.*, 2009): those relying on *pre-existing knowledge resources* (e.g., thesauri, or taxonomies) (Yang and Powers, 2005; Alvarez and Lim, 2007; Hughes and Ramage, 2007), and those based on *distributional properties* (Pereira *et al.*, 1993; Lin, 1998; Chen *et al.*, 2006; Sahami and Heilman, 2006; Pantel *et al.*, 2009).

In the category that relies on existing knowledge sources, the work of Carenini *et al.* (2005) is most related to ours. The authors proposed a method to map feature expressions to a given domain feature taxonomy, using several similarity metrics on WordNet. This work does not use the word distribution information, which is its main weakness because many expressions of the same feature are not synonyms in WordNet as they are domain/application dependent. Dictionaries do not contain domain specific knowledge, for which a domain corpus is needed.

Another related work is distributional similarity, i.e., words with similar meaning tend to appear in similar contexts (Harris, 1968). As such, it fetches the surrounding words as context for each term. Similarity measures such as *Cosine*, *Jaccard*, *Dice*, etc (Lee, 1999), can be employed to compute the similarities between the seeds and other feature expressions. To suit our need, we tested the *k*-means clustering with distributional similarity. However, it does not perform as well as the proposed method.

Recent work also applied topic modeling (e.g., LDA) to solve the problem. Guo *et al.* (2009) proposed a multilevel latent semantic association technique (called *mLSA*) to group product feature expressions, which runs LDA twice. However, *mLSA* is an unsupervised approach. For our evaluation, we still implemented the method and compared it with our SC-EM method.

Our work is also related to constrained clustering (Wagstaff *et al.*, 2001), which uses two forms of constraints, must-link and cannot-link. Must-links state that some data points must be in the same cluster, and cannot-links state that some data points cannot be in the same cluster. In (Andrzejewski *et al.*, 2009), the two constraints are added to LDA, called *DF-LDA*. We show that both these methods do not perform as well as our semi-supervised learning method *SC-EM*.

3 The Proposed Algorithm

Since our problem can be formulated as semi-supervised learning, we briefly describe the setting in our context. Given a set C of classes (our feature groups), we use L to denote the small set of labeled examples (labeled feature expressions or seeds), and U the set of unlabeled examples (unlabeled feature expressions). A classifier is built using L and U to classify every example in U to a class. Several existing algorithms can be applied. In this work, we use EM as it is efficient and it allows prior knowledge to be used easily. Below, we first introduce the EM algorithm that we use, and then present our augmented EM. The constraints and their conflict handling are discussed in Section 4.

3.1 Semi-Supervised Learning Using EM

EM is a popular iterative algorithm for maximum likelihood estimation in problems with missing data. In our case, the group memberships of the unlabeled expressions are considered missing because they come without group labels.

We use the EM algorithm based on naïve Bayesian classification (Nigam *et al.*, 2000). Although it is involved to derive, using it is simple. First, a classifier f is learned using only the labeled data L (Equations 1 and 2). Then, f is applied to assign a probabilistic label to each unlabeled example in U (see Equation 3). Next, a new classifier f is learned using both L and the newly probabilistically labeled unlabeled examples in U_{PL}, again using Equations 1 and 2. These last two steps iterate until convergence.

We now explain the notations in the Equations. Given a set of training documents D, each document d_i in D is considered as an ordered list of words. $w_{d_i,k}$ denotes the k^{th} word in d_i, where each word is from the vocabulary $V=\{w_1, w_2, \ldots, w_{|V|}\}$. $C=\{c_1, c_2, \ldots, c_{|C|}\}$ is the set of pre-defined

$$\mathrm{P}(w_t|c_j) = \frac{1 + \sum_{i=1}^{|D|} N_{ti}\mathrm{P}(c_j|d_i)}{|V| + \sum_{m=1}^{|V|}\sum_{i=1}^{|D|} N_{mi}\mathrm{P}(c_j|d_i)} \quad (1^1)$$

$$\mathrm{P}(c_j) = \frac{1 + \sum_{i=1}^{|D|} \mathrm{P}(c_j|d_i)}{|C| + |D|} \quad (2^1)$$

$$\mathrm{P}(c_j|d_i) = \frac{\mathrm{P}(c_j)\prod_{\mathrm{k}=1}^{|d_i|}\mathrm{P}(w_{d_i,k}|c_j)}{\sum_{\mathrm{r}=1}^{|C|}\mathrm{P}(c_r)\prod_{\mathrm{k}=1}^{|d_i|}\mathrm{P}(w_{d_i,k}|c_r)} \quad (3)$$

classes or groups. N_{ti} is the number of times the word w_t occurs in document d_i.

For our problem, the surrounding words contexts of the labeled seeds form L, while the surrounding words of the non-seed feature expressions form U. When EM converges, the classification labels of the unlabeled feature expressions give us the final grouping. Surrounding words contexts will be discussed in Section 5.

3.2 Proposed Soft-Constrained EM

Although EM can be directly applied to deal with our problem, we can do better. As we discussed earlier, EM only achieves local optimal based on the initialization, i.e., the labeled examples or seeds. We show that natural languages constraints can be used to provide a better initialization, i.e., to add more seeds that are likely to be correct, called *soft-labeled examples* or *soft seeds* (*SL*). Soft-labeled examples are handled differently from the original labeled examples in L. With the soft seeds, we have the proposed soft-constrained EM (called SC-EM).

Compared with the original EM, SC-EM has two main differences:

- Soft constraints are applied to L and U to produce a set *SL* of soft-labeled examples (or soft seeds) to initialize EM in addition to L. *SL* is thus a subset of U. The training set size is increased, which helps produce better results as our experimental results show.
- In the first iteration of EM, soft-labeled examples *SL* are treated in the same way as the labeled examples in L. Thus both *SL* and L are used as labeled examples to learn the initial classifier f_0. However, in the subsequent iterations, *SL* is treated in the same way as any examples in U. That is, the classifier f_x from each iteration x (including f_0) will predict U. After that, a new classifier is built using both L and U_{PL} (which is U with probabilistic labels). Clearly, this implies that the class labels of the examples in *SL* are allowed to change. That is also why we call *SL* the soft-labeled set in contrast to the hard-labeled set L, i.e., the examples in L will not change labels in EM. The reason that *SL* is allowed to change labels/classes is because the constraints can make mistakes. EM may be able to correct some of the mistakes.

[1] Laplace smoothing is used to prevent zero probabilities for infrequently occurring words.

Input:
- Labeled examples L
- Unlabeled examples U

1 Extract *SL* from U using constraints (Section 4);
2 Learn an initial naïve Bayesian classifier f_0 using $L \cup SL$ and Equations 1 and 2;
3 **repeat**
4 // E-Step
5 **for** each example d_i in U (including *SL*) **do**
6 Using the current classifier f_x to compute $\mathrm{P}(c_j|d_i)$ using Equation 3.
7 **end**
8 // M-Step
9 Learn a new naïve Bayesian classifier f_x from L and U by computing $\mathrm{P}(w_t|c_j)$ and $\mathrm{P}(c_j)$ using Equations 1 and 2.
10 **until** the classifier parameters stabilize

Output: the classifier f_x from the last iteration.

Figure 1. The proposed SC-EM algorithm

The detailed algorithm is given in Figure 1. The constraints are discussed in Section 4.

4 Generating SL Using Constraints

As mentioned earlier, two forms of constraints are used to induce the soft-labeled set *SL*. For easy reference, we reproduce them here:

1. Feature expressions sharing some common words are likely to belong to the same group.
2. Feature expressions that are synonyms in a dictionary are likely to belong to one group.

According to the number of words, feature expressions can be categorized into single-word expressions and phrase expressions. They are handled differently. The detailed algorithm is given in Figure 2. In the algorithm, L is the labeled set and U is the unlabeled set. L, in fact, consists of a set of sets, $L = \{L_1, L_2, \ldots, L_{|L|}\}$. Each L_i contains a set of labeled examples (feature expressions) of the i^{th} class (feature group). Similarly, the output set *SL* (the soft-labeled set) also consists of a set of sets, i.e., $SL = \{SL_1, SL_2, \ldots, SL_{|L|}\}$. Each SL_i is a set of soft-labeled examples (feature expressions) of the i^{th} class

(feature group). Thus L_i and SL_i correspond to each other as they represent the original labeled examples and the newly soft-labeled examples of the i^{th} class (or feature group) respectively.

The algorithm basically compares each feature expression u in U (line 1) with each feature expression e (line 4) in every labeled subset L_i (line 2) based on the above two constraints. If any of the constraints is satisfied (lines 5-17), it means that u is likely to belong to L_i (or the i^{th} class or feature group), and it is added to SL_i.

There are conflict situations that need to be resolved. That is, u may satisfy a constraint of more than one labeled sub-set L_i. For example, if u is a single word, it may be synonyms of feature expressions from more than one feature groups. The question is which group it is likely to belong. Further, u may be synonyms of a few single-word feature expressions in L_i. Clearly, u being a synonym of more than one word in L_i is better than it is only the synonym of one word in L_i. Similar problems also occur when u is an element of a feature expression phrase e.

To match u and e, there are a few possibilities. If both u and e are single words (lines 5-6), the algorithm checks if they are synonyms (line 7). The score in line 8 is discussed below. When one of u and e is a phrase, or both of them are phrases, we see whether they have shared words. Again, conflict situations can happen with multiple classes (feature groups) as discussed above. Note that in these cases, we do not use the synonym constraint, which does not help in our test.

Given these complex cases, we need to decide which class that u should be assigned to or should not be assigned to any class (as it does not meet any constraint). We use a score to record the level of satisfaction. Once u is compared with each e in every class, the accumulated score is used to determine which class L_i has the strongest association with u. The class j with the highest score is assigned to u. In other words, u is added to SL_j. Regarding the score value, synonyms gets the score of 1 (line 8), and intersection (shared words) gets the score equal to the size of the intersection (lines 10-17).

```
1  for each feature expression u ∈ U do
2     for each feature group L_i ∈ L do
3        score(L_i) ← 0;
4        for each feature expression e ∈ L_i do
5           if u is a single word expression then
6              if e is a single word expression then
7                 if u and e are synonyms then
8                    score(L_i) ← score(L_i) + 1;
9              else if w ∈ e then  // e is a phrase
10                score(L_i) ← score(L_i) + 1
11          else  // u is a phrase
12             if e is a single word expression then
13                if e ∈ u then  // u is a phrase
14                   score(L_i) ← score(L_i) + 1
15             else
16                s ← e ∩ u;
17                score(L_i) ← score(L_i) + |s|
18    u is added to SL_j s.t. argmax_{L_i} score(L_i)
```

Figure 2. Generating the soft-labeled set SL

5 Distributional Context Extraction

To apply the proposed algorithm, a document d_i needs to be prepared for each feature expression e_i for naïve Bayesian learning. d_i is formed by aggregating the distributional context of each sentence s_{ij} in our corpus that contains the expression e_i. The context of a sentence is the surrounding words of e_i in a text window of [-t, t], including the words in e_i. Given a relevant corpus R, the document d_i for each feature expression e_i in L (or U) is generated using the algorithm in Figure 3. Stopwords are removed.

```
1  for each feature expression e_i in L (or U) do
2     S_i ← all sentences containing e_i in R;
3     for each sentence s_ij ∈ S_i do
4        d_ij ← words in a window of [-t, t] on the left
                and right (including the words in e_i);
5     d_i ← words from all d_ij, j = 1, 2, …, |S_i|;
                // duplicates are kept as it is not union
```

Figure 3. Distributional context extraction

For example, a feature expression from L (or U) is e_i = "*screen*" and there are two sentences in our corpus R that contain "*screen*"

s_{i1} = "*The LCD **screen** gives clear picture*".

s_{i2} = "*The picture on the **screen** is blur*"

We use the window size of [-3, 3]. Sentence s_{i1}, gives us d_{i1} = <LCD, screen, give, clear, picture> as a bag of words. "the" and "is" are removed as stopwords. s_{i2} gives us d_{i2} = <picture, screen, blur>. "on", "the" and "is" are removed as stopwords. Finally, we obtain the document d_i for feature expression e_i as a bag of words:

d_i = <LCD, screen, give, clear, picture, picture, screen, blur>

6 Empirical Evaluation

This section evaluates the SC-EM algorithm and compares it with the main existing methods that can be applied to solve the problem.

6.1 Review Data Sets and Gold Standards

To demonstrate the generality of the proposed method, experiments were conducted using reviews from five domains: *Hometheater, Insurance, Mattress, Car* and *Vacuum*. All the data sets and the *gold standard* feature expressions and groups were from a company that provides opinion mining services. The details of the data sets and the gold standards are given in Table 1.

	Hometheater	Insurance	Mattress	Car	Vacuum
#Sentences	6355	12446	12107	9731	8785
#Reviews	587	2802	933	1486	551
#Feature expressions	237	148	333	317	266
#Feature groups	15	8	15	16	28

Table 1. Data sets and gold standards

6.2 Evaluation Measures

Since SC-EM is based on semi-supervised learning, we can use classification accuracy to evaluate it. We can also see it as clustering with initial seeds. Thus we also use clustering evaluation methods. Given gold standards, two popular clustering evaluation measures are *Entropy* and *Purity* (Liu, 2006). As *accuracy* is fairly standard, we will not discuss it further. Below, we briefly describe entropy and purity.

Given a data set *DS*, its gold partition is $G = \{g_1,\ldots, g_j,\ldots, g_k\}$, where k is the known number of clusters. The groups partition *DS* into k disjoint subsets, $DS_1,\ldots, DS_i, \ldots, DS_k$.

Entropy: For each resulting cluster, we can measure its entropy using Equation 4, where $\mathrm{P}_i(g_j)$ is the proportion of g_j data points in DS_i. The total entropy of the clustering (considering all clusters) is calculated by Equation 5.

$$entropy(DS_i) = -\sum_{j=1}^{k} \mathrm{P}_i(g_j) log_2 \mathrm{P}_i(g_j) \quad (4)$$

$$entropy_{total} = \sum_{i=1}^{k} \frac{|DS_i|}{|DS|} entropy(DS_i) \quad (5)$$

Purity: Purity measures the extent that a cluster contains only data from one gold-partition. Each cluster's purity is computed by Equation 6, and the total purity of the whole clustering is computed with Equation 7.

$$purity(DS_i) = \max_j \mathrm{P}_i(g_j) \quad (6)$$

$$purity_{total} = \sum_{i=1}^{k} \frac{|DS_i|}{|DS|} purity(DS_i) \quad (7)$$

In testing, the unlabeled set *U* is also our test set. This is justified because our purpose is to assign unlabeled data to appropriate groups.

6.3 Baseline Methods and Settings

The proposed ***SC-EM*** method is compared with a set of existing methods, which can be categorized into unsupervised and semi-supervised methods. We list the *unsupervised* methods first.

LDA: *LDA* is a popular topic modeling method (see Section 2). Given a set of documents, it outputs groups of terms of different topics. In our case, each feature expression is a term, and the documents refer to the distributional contexts of each feature expressions (see Section 5).

mLSA: This is a state-of-the-art unsupervised method for solving the problem. It is based on LDA, and has been discussed in related work.

Kmeans: This is the *k*-means clustering method (MacQueen, 1966) based on distributional similarity with cosine as the similarity measure.

In the *semi-supervised* category, the methods are further classified into un-constrained, hard-constrained, and soft-constrained methods.

For the *un-constrained* subclass (no constraints are used), we have the following:

LDA*(*L, H*)**: This method is based on ***LDA, but the labeled examples *L* are used as seeds for each group/topic. All examples in *L* will always stay in the same topic. We call this hard initialization (*H*). *L* is handled similarly below.

DF-LDA*(*L, H*)**. *DF-LDA* is the *LDA* method (Andrzejewski *et al.*, 2009) that takes must-links and cannot-links. Our *L* set can be expressed as a combination of must-links and cannot-links. Unfortunately, only must-links can be used because the number of cannot-links is huge and crashes the system. For example, for the car data, the number of cannot-links is 194,400 for 10% labeled data (see Section 6.4) and for 20% it is 466,560,000. DF-LDA also has a parameter η controlling the link strength, which is set very high (=1000) to reflect the hard initialization. We did not use *DF-LDA* in the unsupervised subclass above as without constraints it reduces to ***LDA.

***Kmeans*(*L, H*)**: This method is based on *Kmeans*, but the clusters of the labeled seeds are fixed at the initiation and remain unchanged.

***EM*(*L, H*)**: This is the original EM for semi-supervised learning. Only the labeled examples are used as the initial seeds.

For the *hard-constrained* (*H*) subclass (our

two constraints are applied and cannot be violated), we have the following methods (*LC* is *L* plus *SL* produced by the constraints (C):

***Rand*(*LC, H*):** This is an important baseline. It shows whether the constraints alone are sufficient to produce good results. That is, the final result is the expanded seeds *SL* plus the rest of *U* assigned randomly to different groups.

***LDA*(*LC, H*)**: It is similar to *LDA*(*L,H*), but both the initial seeds *L* and the expanded seeds *SL* are considered as labeled examples. They also stay in the same topics/groups in the process. Note that although *SL* is called a set of soft-labeled examples (seeds) in the proposed algorithm, they are treated as hard-labeled examples here just for experimental comparison.

***DF-LDA*(*LC, H*)**: This is *DF-LDA* with both *L* and *SL* expressed as must-links. Again, a large η (= 1000) is used to make sure that must-links for *L* and *SL* will not be violated.

***Kmeans*(*LC,H*)**: It is similar to *Kmeans*(*L,H*), but both *L* and *SL* stay in their assigned clusters.

***EM*(*LC, H*)**: It is similar to *SC-EM*, but *SL* is added to the labeled set *L*, and their classes are not allowed to change in the EM iterations.

For the *soft-constrained* (*S*) subclass, our two constraints can be violated. Initially, both the initial seeds *L* and the expanded seeds *SL* are considered as labeled data, but subsequently, only *L* is taken as the labeled data (i.e., staying in the same classes). The algorithm will re-estimate the label of each feature expression in *SL*. This subclass has the following methods:

***LDA*(*LC, S*)**: This is in contrast to *LDA*(*LC, H*). It allows the *SL* set to change topics/groups.

***Kmeans*(*LC, S*)**: This is in contrast to *Kmeans*(*LC, H*).

A soft *DF-LDA* is not included here because different η values give different results, and they are generally worse than *DF-LDA*(*LC, H*).

For all *LDA* based methods, the topic modeling parameters were set to their default values. The number of iteration is 1000. We used the *LDA* in MALLET[2], and modified it to suit different *LDA*-based methods except *DF-LDA*, which was downloaded from its authors' website[3]. We implemented *mLSA*, *Kmeans* and changed EM[4] to take soft seeds. For all *Kmeans* based methods, the distance function is the cosine similarity.

[2] http://mallet.cs.umass.edu/
[3] http://pages.cs.wisc.edu/~andrzeje/research/df_lda.html
[4] http://alias-i.com/lingpipe/

6.4 Evaluation Results

We now compare the results of *SC-EM* and the 14 baseline methods. To see the effects of different numbers of labeled examples (seeds), we experimented with 10%, 20%, 30%, 40%, and 50% of the feature expressions from the gold standard data as the labeled set *L*, and the rest as the unlabeled set *U*. All labeled data were selected randomly. For each setting, we run the algorithms 30 times and report the average results. Due to space limitations, we can only show the detailed *purity* (Pur), *entropy* (Ent) and *accuracy* (Acc) results for 30% as the labeled data (70% as unlabeled) in Table 2. For the other proportions of labeled data, we summarize them in Table 3. Each result in Table 3 is thus the average of the 5 data sets. All the results were obtained from the unlabeled set *U*, which was our test set. For entropy, the smaller the value is the better, but for purity and accuracy, the larger the better. For these experiments, we used the window size $t = 5$. Section 6.5 studies the effects of window sizes.

Tables 2 and 3 clearly show that the proposed algorithm (***SC-EM***) outperforms all 14 baseline methods by a large margin on every dataset. In detail, we observe the following:

- *LDA*, *mLSA* and *Kmeans* with no seeds (labeled data) perform the worst. Seeds help to improve the results, which is intuitive. Without seeds, *DF-LDA* is the same as *LDA*.
- *LDA* based methods seems to be the weakest. *Kmeans* based methods are slightly better, but EM based methods are the best. This clearly indicates that classification (EM) performs better than clustering. Comparing *DF-LDA* and *Kmeans*, their results are similar.
- For *LDA*, and *Kmeans*, hard-constrained methods (i.e., *LDA*(*L, H*), and *Kmeans*(*L, H*)) perform better than soft-constrained methods (i.e., *LDA*(*LC, S*) and *Kmeans*(*LC, S*)). This indicates that soft-constrained versions may change some correctly constrained expressions into wrong groups. However, for the EM based methods, the soft-constrained method (*SC-EM*) performs markedly better than the hard-constrained version (*EM*(*LC, H*)). This indicates that Bayesian classifier used in EM can take advantage of the soft constraints and correct some wrong assignments made by constraints. Much weaker results of *Rand*(*LC, H*) than *SC-EM* in different settings show that

Methods	Hometheater			Insurance			Mattress			Car			Vacuum		
	Acc	Pur	Ent	Acc	Pur	Ent	Acc	Pur	Ent	Acc	Pur	Ent	Acc	Pur	Ent
LDA	0.06	0.31	2.54	0.11	0.36	2.24	0.05	0.32	2.57	0.06	0.37	2.39	0.03	0.36	2.09
mLSA	0.06	0.31	2.53	0.14	0.38	2.19	0.06	0.34	2.55	0.09	0.37	2.40	0.03	0.37	2.11
Kmeans	**0.21**	**0.42**	**2.14**	**0.25**	**0.45**	**1.90**	**0.15**	**0.39**	**2.32**	**0.25**	**0.44**	**2.16**	**0.24**	**0.47**	**1.78**
LDA(L, H)	0.10	0.32	2.50	0.16	0.37	2.22	0.10	0.34	2.57	0.19	0.39	2.36	0.10	0.39	2.09
DF-LDA(L, H)	0.27	0.37	2.32	0.25	0.41	2.00	0.19	0.39	2.35	0.28	0.45	2.15	0.31	0.40	1.98
Kmeans(L, H)	0.20	0.42	2.12	0.25	0.43	1.92	0.17	0.42	2.26	0.27	0.48	2.04	0.20	0.48	1.76
EM(L, H)	**0.48**	**0.50**	**1.93**	**0.50**	**0.53**	**1.69**	**0.52**	**0.56**	**1.87**	**0.56**	**0.58**	**1.80**	**0.49**	**0.52**	**1.79**
Rand(CL, H)	0.41	0.46	2.07	0.40	0.46	1.94	0.40	0.47	2.07	0.34	0.41	2.31	0.39	0.52	1.59
LDA(CL, H)	0.44	0.50	1.96	0.42	0.48	1.89	0.42	0.49	1.97	0.44	0.52	1.87	0.43	0.55	1.48
DF-LDA(CL, H)	0.35	0.49	1.86	0.33	0.49	1.71	0.23	0.39	2.26	0.34	0.51	1.88	0.37	0.52	1.58
Kmeans(CL, H)	0.49	0.55	1.70	0.48	0.55	1.62	0.44	0.51	1.91	0.47	0.54	1.80	0.44	0.58	1.42
EM(CL, H)	**0.59**	**0.60**	**1.62**	**0.58**	**0.60**	**1.46**	**0.56**	**0.59**	**1.74**	**0.62**	**0.64**	**1.54**	**0.55**	**0.60**	**1.44**
LDA(CL, S)	0.24	0.35	2.44	0.27	0.40	2.14	0.23	0.37	2.44	0.27	0.41	2.33	0.23	0.41	2.01
Kmeans(CL, S)	0.33	0.46	2.04	0.34	0.45	1.90	0.25	0.43	2.20	0.29	0.47	2.07	0.37	0.50	1.68
SC-EM	**0.67**	**0.68**	**1.30**	**0.66**	**0.68**	**1.18**	**0.68**	**0.70**	**1.27**	**0.70**	**0.71**	**1.24**	**0.67**	**0.68**	**1.18**

Table 2. Comparison results (L = 30% of the gold standard data)

Methods	Acc					Pur					Ent				
	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%
LDA	0.07	0.07	0.06	0.06	0.08	0.33	0.33	0.34	0.35	0.38	2.50	2.44	2.37	2.28	2.11
mLSA	0.07	0.07	0.08	0.07	0.07	0.34	0.35	0.35	0.37	0.38	2.48	2.42	2.36	2.26	2.12
Kmeans	**0.22**	**0.23**	**0.22**	**0.22**	**0.22**	**0.42**	**0.43**	**0.44**	**0.44**	**0.46**	**2.16**	**2.11**	**2.06**	**1.98**	**1.86**
LDA(L, H)	0.10	0.10	0.13	0.14	0.15	0.34	0.34	0.36	0.37	0.39	2.48	2.43	2.35	2.25	2.11
DF-LDA(L, H)	0.23	0.25	0.26	0.27	0.30	0.41	0.40	0.41	0.41	0.44	2.23	2.23	2.16	2.10	1.94
Kmeans(L, H)	0.13	0.16	0.22	0.24	0.28	0.42	0.43	0.45	0.45	0.48	2.15	2.11	2.02	1.95	1.79
EM(L, H)	**0.35**	**0.44**	**0.51**	**0.55**	**0.58**	**0.43**	**0.49**	**0.54**	**0.57**	**0.61**	**2.22**	**1.99**	**1.81**	**1.65**	**1.49**
Rand(CL, H)	0.28	0.35	0.39	0.42	0.45	0.39	0.43	0.47	0.50	0.54	2.33	2.15	2.00	1.82	1.63
LDA(CL, H)	0.31	0.38	0.43	0.46	0.49	0.43	0.47	0.51	0.54	0.58	2.16	1.99	1.83	1.69	1.49
DF-LDA(CL, H)	0.32	0.33	0.33	0.34	0.36	0.49	0.50	0.48	0.48	0.48	1.90	1.85	1.86	1.83	1.82
Kmeans(CL, H)	0.33	0.41	0.46	0.49	0.52	0.47	0.51	0.55	0.57	0.61	1.98	1.82	1.69	1.56	1.42
EM(CL, H)	**0.44**	**0.54**	**0.58**	**0.61**	**0.64**	**0.49**	**0.57**	**0.61**	**0.64**	**0.67**	**1.98**	**1.72**	**1.56**	**1.40**	**1.25**
LDA(CL, S)	0.17	0.21	0.25	0.30	0.34	0.34	0.36	0.39	0.42	0.46	2.47	2.37	2.27	2.09	1.87
Kmeans(CL, S)	0.23	0.28	0.32	0.36	0.42	0.43	0.44	0.46	0.48	0.51	2.15	2.08	1.98	1.86	1.70
SC-EM	**0.45**	**0.58**	**0.68**	**0.75**	**0.81**	**0.50**	**0.61**	**0.69**	**0.76**	**0.82**	**1.95**	**1.56**	**1.24**	**0.94**	**0.69**

Table 3. Influence of the seeds' proportion (which reflects the size of the labeled set L)

constraints alone (i.e., synonyms and sharing of words) are far from sufficient. EM can improve it considerably.

- Comparing EM based methods, we can see that soft seeds in *SL* make a big difference for all data sets. *SC-EM* is clearly the best.
- As the number of labeled examples increases (from 10% to 50%), the results improve for every method (except those for *DF-LDA*, which does not change much).

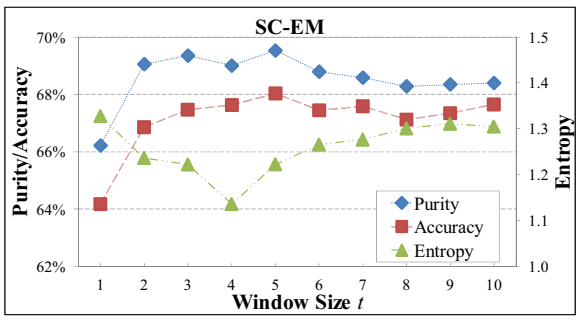


Figure 4. Influence of context window size

6.5 Varying the Context Window Size

We varied the text window size t from 1 to 10 to see how it impacts on the performance of *SC-EM*. The results are given in Figure 4 (they are averages of the 5 datasets). Again for purity and accuracy, the greater the value the better, while for entropy it is the opposite. It is clear that the window sizes of 2~6 produce similar good results. All evaluations reported above used t = 5.

7 Conclusion

This paper proposed the task of feature grouping in a semi-supervised setting. It argued that some form of supervision is needed for the problem because its solution depends on the user application needs. The paper then proposed to use the EM algorithm to solve the problem, which was improved by considering two soft constraints. Empirical evaluations using 5 real-life data sets show that the proposed method is superior to 14 baselines. In our future work, we will focus on further improving the accuracy.

References

Agirre E., E. Alfonseca, K. Hall, J. Kravalova, M. Pa ca and A. Soroa 2009. *A study on similarity and relatedness using distributional and WordNet-based approaches*. Proceedings of ACL.

Alvarez M. and S. Lim 2007. *A Graph Modeling of Semantic Similarity between Words*. Proceeding of the Conference on Semantic Computing.

Andrzejewski D., X. Zhu and M. Craven 2009. *Incorporating domain knowledge into topic modeling via Dirichlet forest priors*. Proceedings of ICML.

Blei D., A. Y. Ng and M. I. Jordan 2003. "*Latent Dirichlet Allocation*." JMLR 3: 993-1022.

Carenini G., R. Ng and E. Zwart 2005. *Extracting knowledge from evaluative text*. Proceedings of International Conference on Knowledge Capture.

Chen H., M. Lin and Y. Wei 2006. *Novel association measures using web search with double checking*. Proceedings of ACL.

Dempster A., N. Laird and D. Rubin 1977. "*Maximum likelihood from incomplete data via the EM algorithm*." Journal of the Royal Statistical Society 39(1): 1-38.

Guo H., H. Zhu, Z. Guo, X. Zhang and Z. Su 2009. *Product feature categorization with multilevel latent semantic association*. Proc. of CIKM.

Harris Z. S. 1968. *Mathematical structures of language*. New York, Interscience Publishers.

Hu M. and B. Liu 2004a. *Mining and summarizing customer reviews*. Proceedings of SIGKDD.

Hu M. and B. Liu 2004b. *Mining Opinion Features in Customer Reviews*. Proceedings of AAAI.

Hughes T. and D. Ramage 2007. *Lexical semantic relatedness with random graph walks*. EMNLP.

Jin W., H. Ho and R. Srihari 2009. *OpinionMiner: a novel machine learning system for web opinion mining and extraction*. Proceedings of KDD.

Joachims T. 1999. *Transductive inference for text classification using support vector machines*. Proceedings of ICML.

Kim S. and E. Hovy 2006. *Extracting opinions, opinion holders, and topics expressed in online news media text*. Proceedings of EMNLP.

Kobayashi N., K. Inui and Y. Matsumoto 2007. *Extracting aspect-evaluation and aspect-of relations in opinion mining*. Proceedings of EMNLP.

Ku L., H. Ho and H. Chen 2009. "*Opinion mining and relationship discovery using CopeOpi opinion analysis system*." Journal of the American Society for Information Science and Technology 60(7): 1486-1503.

Lee L. 1999. *Measures of distributional similarity*, Proceedings of ACL.

Lin D. 1998. *Automatic retrieval and clustering of similar words*, Proceedings of ACL.

Liu B. 2006. *Web data mining; Exploring hyperlinks, contents, and usage data*, Springer.

Liu B. 2010. *Sentiment Analysis and Subjectivity*. *Handbook of Natural Language Processing* N. Indurkhya and F. J. Damerau.

MacQueen J. 1966. *Some methods for classification and analysis of multivariate observations*. Proc. of Symposium on Mathematical Statistics and Probability.

Mei Q., X. Ling, M. Wondra, H. Su and C. Zhai 2007. *Topic sentiment mixture: modeling facets and opinions in weblogs*. Proceedings of WWW.

Nigam K., A. McCallum, S. Thrun and T. Mitchell 2000. "*Text classification from labeled and unlabeled documents using EM.*" Machine Learning 39(2).

Pang B. and L. Lee 2008. "*Opinion mining and sentiment analysis*." Foundations and Trends in Information Retrieval 2(1-2): 1-135.

Pantel P., E. Crestan, A. Borkovsky, A. Popescu and V. Vyas 2009. *Web-scale distributional similarity and entity set expansion*. EMNLP.

Pereira F., N. Tishby and L. Lee 1993. *Distributional clustering of English words*. Proceedings of ACL.

Popescu A.-M. and O. Etzioni 2005. *Extracting Product Features and Opinions from Reviews*. EMNLP.

Sahami M. and T. Heilman 2006. *A web-based kernel function for measuring the similarity of short text snippets*. Proceedings of WWW.

Stoyanov V. and C. Cardie 2008. *Topic identification for fine-grained opinion analysis*. COLING.

Wagstaff K., C. Cardie, S. Rogers and S. Schroedl 2001. *Constrained k-means clustering with background knowledge*. In Proceedings of ICML.

Yang D. and D. Powers 2005. *Measuring semantic similarity in the taxonomy of WordNet*, Proceedings of the Australasian conference on Computer Science.

Forest-guided Supertagger Training

Yao-zhong Zhang † **Takuya Matsuzaki** † **Jun'ichi Tsujii**†‡§

† Department of Computer Science, University of Tokyo
‡ School of Computer Science, University of Manchester
§National Centre for Text Mining
{yaozhong.zhang, matuzaki, tsujii}@is.s.u-tokyo.ac.jp

Abstract

Supertagging is an important technique for deep syntactic analysis. A supertagger is usually trained independently of the parser using a sequence labeling method. This presents an inconsistent training objective between the supertagger and the parser. In this paper, we propose a forest-guided supertagger training method to alleviate this problem by incorporating global grammar constraints into the supertagging process using a CFG-filter. It also provides an approach to make the supertagger and the parser more tightly integrated. The experiment shows that using the forest-guided trained supertagger, the parser got an absolute 0.68% improvement from baseline in F-score for predicate-argument relation recognition accuracy and achieved a competitive result of 89.31% with a faster parsing speed, compared to a state-of-the-art HPSG parser.

1 Introduction

Deep syntactic analysis by lexicalized grammar parsing, which provides linguistic-rich information for many NLP tasks, has recently received more and more attention from the NLP community. To use a deep parser in real large-scale applications, speed is an important issue to take into consideration. Supertagging is one of the speed-up technique widely used for lexicalized grammar parsing. A supertagger is used to limit the number of plausible lexical entries fed to the parser, this can greatly reduce the search space for the parser. Supertagging was first proposed for Lexicalized Tree Adjoining Grammar (LTAG) (Bangalore and Joshi, 1999), and then successfully applied to Combinatory Categorial Grammar (CCG) (Clark, 2002) and Head-driven Phrase Structure Grammar (HPSG) (Ninomiya et al., 2006). In addition, supertags can also be used for other NLP tasks besides parsing, such as semantic role labeling (Chen and Rambow, 2003) and machine translation (Birch et al., 2007; Hassan et al., 2007) to utilize syntactic information in the supertags.

In lexicalized grammar parsing, supertagging is usually treated as a sequence labeling task independently trained from the parser. Previous research (Clark, 2002) showed that even a pointwise classifier not considering context edge features is effective when used as a supertagger. To make up for the insufficient accuracy as a single-tagger, more than one supertag prediction is reserved and the parser takes the burden of resolving the rest of the supertag ambiguities.

A non-trivial problem raised by the separate training of the supertagger is that the prediction score provided by the supertagger might not be suitable for direct use in the parsing process, since a separately trained supertagger that does not take into account grammar constraints has a training objective which is inconsistent with the parser. Although the scores provided by the supertagger can be ignored (e.g., in some CCG parsers), this may also discard some useful information for effective beam search and accurate disambiguation.

Based on this observation, we assume that considering global grammar constraints during the supertagger training process would make the supertagger and the parser more tightly integrated.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1281–1289,
Beijing, August 2010

In this paper, we propose an on-line forest-guided training method for a supertagger to make the training objective of a supertagger more closely related to the parsing task. We implemented this method on a large-scale HPSG grammar. We used a CFG grammar to approximate the original HPSG grammar in the supertagging stage and applied best-first search to select grammar-satisfying supertag sequences for the parameter updating. The experiments showed that the HPSG parser is improved by considering structure constraints in the supertagging training process. For the standard test set (Penn Treebank Section 23), we accomplished an absolute 0.68% improvement from baseline in F-score for predicate-argument relation recognition and got a competitive result of 89.31% with a faster parsing speed, compared to a state-of-the-art HPSG parser.

The remainder of the paper is organized as follows: in section 2 we provide the necessary background regarding HPSG parsing. In section 3, we introduce the on-line forest-guided supertagger training method. Section 4 shows the experiment results and the related analysis. Section 5 compares the proposed approach with related work and section 6 presents our conclusions and future work.

2 Background

2.1 Statistical HPSG Parsing

HPSG (Pollard and Sag, 1994) is a lexicalist grammar framework. In HPSG, a large number of lexical entries are used to express word-specific characteristics, while only a small number of rule schemata are used to describe general construction rules. Typed feature structures named "signs" are used to represent both lexical entries and phrasal constituents. A classic efficient statistical HPSG parsing process is depicted in Figure 1. Given a word and part-of-speech sequence (w, p) as input, the first step (called "supertagging") in HPSG parsing is to assign possible lexical entries. In practice, for each word, more than one supertag is reserved for the parser. Then, the parser searches the given lexical entry space to construct a HPSG tree using the rule schemata to combine possible signs. Constituent-based methods and transition-based methods can be used for tree structure disambiguation. This parsing framework using supertagging is also used in other lexicalized grammars, such as LTAG and CCG.

2.2 HPSG Supertagging

Like other lexicalized grammar, the lexical entries defined in HPSG are referred to as "supertags". For example, the word "like" is assigned a lexical entry for transitive verbs in non-3rd person present form, which indicates that the head syntactic category of "like" is verb and it has an NP subject and an NP complement. With such fine-grained grammatical type distinctions, the number of supertags is very large. Compared to the 45 part-of-speech (POS) tags defined in the PennTreebank, the HPSG grammar we used contains 2,308 supertags. The large number and the complexity of the supertags makes supertagging harder than the POS tagging task.

Supertagging can be formulated as a sequence labeling task. Here, we follow the definition of Collins' perceptron (Collins, 2002). The training objective of supertagging is to learn the mapping from a POS-tagged word sentence $\boldsymbol{w} = (w_1/p_1, ..., w_n/p_n)$ to a sequence of supertags $\boldsymbol{s} = (s_1, ..., s_n)$. We use function $\boldsymbol{GEN(w)}$ to indicate all candidates of supertag sequences given input $\boldsymbol{w}$. Feature function $\boldsymbol{\Phi}$ maps a sample $(\boldsymbol{w}, \boldsymbol{s})$ to a point in the feature space R^d. $\boldsymbol{\theta}$ is the vector of feature weights. Given an input $\boldsymbol{w}$, the most plausible supertag sequence is found by the prediction function defined as follows:

$$F(\boldsymbol{w}) = \underset{\boldsymbol{s} \in \boldsymbol{GEN(w)}}{argmax} \ \theta \cdot \Phi(\boldsymbol{w}, \boldsymbol{s}) \qquad (1)$$

2.3 CFG-filtering

CFG-filtering (Kiefer and Krieger, 2000) is a technique to find a superset of (packed) HPSG parse trees that satisfy the constraints in a grammar. A CFG that approximates the original HPSG grammar is used for efficiently finding such trees without doing full-fledged HPSG parsing that is computationally demanding because the schema application involves unification operations among large feature structures (signs). The number of possible signs is infinite in general and hence

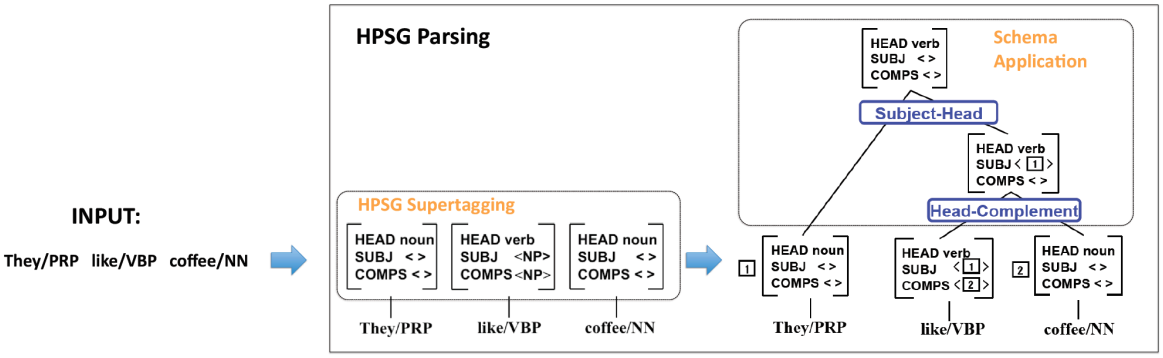


Figure 1: HPSG parsing for the sentence "They like coffee."

some features (e.g., the number agreement feature) are ignored in the approximating CFG so that the set of possible signs can be approximated by a finite set of non-terminal symbols in the CFG. By this construction, some illegal trees may be included in the set of trees licensed by the approximating CFG, but none of the well-formed trees (i.e., those satisfying all constraints in the grammar) are excluded by the approximation. We use the algorithm described by Kiefer and Krieger (2000) to obtain the approximating CFG for the original HPSG. The technical details regarding the algorithm can be found in Kiefer and Krieger (2000).

3 Forest-guided Training for Supertagging

3.1 Motivation

In lexicalized grammar parsing, a parser aims to find the most plausible syntactic structure for a given sentence based on the supertagging results. One efficient parsing approach is to use prediction scores provided by the supertagger. Usually, the supertagger is trained separately from the structure disambiguation in a later stage. This pipeline parsing strategy poses a potential problem in that the training objective of a supertagger can deviate from the final parser, if the global grammar constraints are not considered. For example, the supertag predictions for some words can contribute to high supertagging accuracy, but cause the parser to fail. Therefore, considering the global grammar constraints in the supertagging training stage can make the supertagger and the parser more tightly related, which will contribute towards the performance of the parser.

Algorithm 1: Forest-guided supertagger training

Input: Training Sample $(w_i, s_i)_{i=1,...,N}$,
Number of iterations T

$\theta \leftarrow (0, ..., 0)$, $\theta_{sum} \leftarrow (0, ..., 0)$
for $iterNum \leftarrow 1$ to T **do**
  for i $\leftarrow 1$ to N **do**
    Generate supertag lattice using the point-wise classifier with current θ
    Select $\hat{s}_i$ from the lattice which **can construct a tree** with largest sequence score
    if(No $\hat{s}_i$ satisfied grammar constraints)
    $\hat{s}_i \leftarrow \arg\max_{s \in \mathbf{GEN}(w_i)} \theta_i \cdot \Phi(w_i, s_i)$
    if $\hat{s}_i \neq s_i$ then
      $\theta_{i+1} \leftarrow \theta_i + \Phi(w_i, s_i) - \Phi(w_i, \hat{s}_i)$
      $\theta_{sum} \leftarrow \theta_{sum} + \theta_{i+1}$
Return: θ_{sum}/NT

3.2 Training Algorithm

Based on the motivation above, we propose a forest-guided supertagger training method to make the supertagger more tightly integrated with the parser. This method is based on the averaged perceptron training algorithm. The training process is given in Algorithm 1.

The most important difference of the proposed algorithm compared to the traditional supertagger training method is that the current best-scored supertag sequence is searched only within the space of the supertag sequences that are allowed by the grammar. As for whether the grammar

constraints are satisfied, we judge it by whether a possible syntactic tree can be constructed using the given supertag sequence. We do not require the constructed syntactic tree to be identical to the gold tree in the corpus. For this reason we call it "forest-guided".

In the forest-guided training of the supertagger, an approximating CFG is used to filter out the supertag sequences from which no well-formed tree can be built. It is implemented as a best-first CFG parser wherein the score of a constituent is the score of the supertag (sub-)sequence on the fringe of the constituent, which is calculated using the current value of the parameters. Note that the best-first parser can find the best-scored supertag sequence very efficiently given proper scoring for the candidate supertag set for each token; this is actually the case in the course of training except for the initial phase of the training, wherein the parameter values are not well-tuned. The efficiency is due to the sparseness of the approximating CFG (i.e., the production rule set includes only a tiny fraction of the possible parent-children combinations of symbols) and highest-scored supertags often have a well-formed tree on top of them.

As is clear from the above description, the use of CFG-filter in the forest-guided training of the supertagger is not essential but is only a subsidiary technique to make the training faster. The improvement by the forest-guided training should however depend on whether the CFG approximation is reasonably tight or not. Actually, we managed to obtain a manageable size out of a CFG grammar, which includes 80 thousand nonterminal symbols and 10 million rules, by eliminating only a small number of features (semantics, case and number agreement, and fine distinctions in nouns, adjectives and complementizers). We thus believe that the approximation is fairly tight.

This training algorithm can also be explained in a search-based learning framework (Hal Daumé III and Daniel Marcu, 2005). In this framework, the objective of learning is to optimize the $\boldsymbol{\theta}$ for the enqueue function to make the good hypotheses rank high in the search queue. The rank score $\boldsymbol{r}$ consists of two components: path score $\boldsymbol{g}$ and heuristic score $\boldsymbol{h}$. In the forest-guided training method, $\boldsymbol{r}$ can be rewritten as follows:

$$\begin{aligned} r &= g + h \\ &= \theta \cdot \Phi(x, \hat{y}) + 1_{[Tree(\hat{y})]} * Penalty \end{aligned} \quad (2)$$

The heuristic part $\boldsymbol{h}$ checks whether the supertag candidate sequence satisfies the grammar constraints: if no CFG tree can be constructed, -∞ penalty is imposed to the candidate sequence in the forest-guided training method.

4 Experiments

We mainly evaluated the proposed forest-guided supertagger training method on HPSG parsing. Supertagging accuracy[1] using different training methods was also investigated.

4.1 Corpus Description

The HPSG grammar used in the experiments is Enju version 2.3[2]. It is semi-automatically converted from the WSJ portion of PennTreebank (Miyao, 2006). The grammar consists of 2,308 supertags in total. Sections 02-21 were used to train different supertagging models and the HPSG parser. Section 22 and section 23 were used as the development set and the test set respectively. We evaluated the HPSG parser performance by labeled precision (LP) and labeled recall (LR) of predicate-argument relations of the parser's output as in previous works (Miyao, 2005). All experiments were conducted on an AMD Opteron 2.4GHz server.

Template Type	Template
Word	w_i,w_{i-1},w_{i+1}, $w_{i-1}\&w_i$, $w_i\&w_{i+1}$
POS	p_i, p_{i-1}, p_{i-2}, p_{i+1}, p_{i+2}, $p_{i-1}\&p_i$, $p_{i-2}\&p_{i-1}$, $p_{i-1}\&p_{i+1}$, $p_i\&p_{i+1}$, $p_{i+1}\&p_{i+2}$
Word-POS	$p_{i-1}\&w_i$, $p_i\&w_i$, $p_{i+1}\&w_i$

Table 1: Feature templates used for supertagging models.

[1] "UNK" supertags are ignored in evaluation as in previous works.

[2] http://www-tsujii.is.s.u-tokyo.ac.jp/enju/index.html

4.2 Baseline Models and Settings

We used a point-wise averaged perceptron (PW) to train a baseline supertagger. Point-wise classifiers have been reported to be very effective and with competitive results for the supertagging task (Clark, 2002; Zhang et al., 2009). The number of training iterations was set to 5. The features used in the supertaggers are described in Table 1. For comparison, these features are identical to the features used in the previous works (Matsuzaki et al., 2007; Ninomiya et al., 2007). To make the training efficient, we set the default chart size limit for the forest-guided supertagger training to be 20k by tuning it on the development set.

We combined the supertagger trained under forest-guidance with a supertagging-based HPSG parser (Matsuzaki et al., 2007) and evaluated the contribution of the improved supertagger training procedure for the final HPSG parsing by the accuracy of the predicate-argument relations output of the parser. The parser crucially depends on the supertagger's performance in that it outputs the first well-formed tree successfully constructed on the highest scored supertag sequence. The highest-scored supertag sequences are enumerated one by one in descending order in regards to their score. The enumeration is actually implemented as n-best parsing on the supertag candidates using an approximating CFG. The HPSG tree construction on a supertag sequence is done using a shift-reduce style parsing algorithm equipped with a classifier-based action selection mechanism.

The automatically assigned POS tags were given by a maximum entropy tagger with roughly 97% accuracy.

4.3 Supertagging Results

Although we mainly focused on improving the final HPSG parsing performance through the improved supertagger training, it is also very interesting to investigate the supertagger performance using different training methods. To evaluate the forest-guided training method for a supertagger, we also need to incorporate structure constraints in the test stage. To make fair comparisons, for the averaged perceptron trained supertagger we also add structure constraints in its testing.

	Model Name	Acc%
auto-POS	FT+CFG	**92.77**
	PW+CFG	92.47
	PW	91.14
	ME	91.45
gold-POS	FT+CFG	**93.98**
	PW+CFG	93.70
	PW	92.48
	ME	92.78

Table 2: Supertagging results in section 23. "FT" represents the forest-guided trained supertagger. "PW" is the baseline average perceptron trained supertagger. "ME" is the supertagger trained by using the maximum entropy method. "+CFG" indicates the use of the CFG-filter for the supertagger results. The accuracy of automatically assigned POS tags in this section is 97.39%.

For simplicity, throughout this paper, we call the forest-guided trained supertagger "FT" in short, while the "PW" is used to represent the baseline point-wise averaged perceptron supertagger. "ME" is the re-implemented maximum entropy supertagger described in Matsuzaki et al. (2007).

For the PW supertagger, the performance was roughly 0.3% below the ME supertagger. Similar results were reported by Zhang et al. (2009), which used a Bayes point machine to reduce the gap between the averaged perceptron supertagger and the maximum entropy supertagger. Although we expected the ME supertagger using CFG-filter to give better results than the PW supertagger, implementing forest-guided supertagger training in a maximum entropy framework is different and more sophisticated than the current on-line training method. Considering that the performance of the PW supertagger and the ME supertagger were at a similar level, we chose the PW supertagger as our baseline.

We used a CFG-filter to incorporate global grammar constraints into both the training and the testing phase. Compared to the PW supertagger, the PW+CFG supertagger incorporated global grammar constraints only in the test phase, while for the FT+CFG supertagger, the global grammar constraints were incorporated both in

Training Method \ Iter NUM	1	2	3	4	5	Total Time
FT	6684s	4189s	3524s	3285s	3086s	$\approx$ 5.8h
PW	99s	116s	117s	117s	117s	$\approx$ 10 min
ME	/					$\approx$ 3h

Table 3: Supertagger training time on section 02-21. "FT" and "PW" represent forest-guided training and point-wise averaged perceptron training separately. "ME" is the point-wise maximum entropy training reported in Matsuzaki et al. (2007).

the training and the testing stage. The supertagging accuracy for different models is shown in Table 2. Firstly, incorporating grammar constraints only in the testing phase (PW+CFG) gave an absolute 1.22% (gold POS) and 1.33% (auto POS) increase in F-score compared to the PW supertagger. Secondly, incorporating grammar constraints into both the training and the testing stage (FT+CFG) gave an additional 0.28% (gold POS) and 0.3% (auto POS) improvement over the PW+CFG supertagger with p-values 0.0018 (gold POS) and 0.0016 (auto POS).

This also indicates that the supertagger and the parser are closely related to each other. The original motivation for supertagging is using simple models to resolve lexical ambiguities, which can efficiently reduce the search space of the parser. A better supertagger can contribute to more efficient and more accurate lexicalized grammar parsing. Actually, a supertagger can act as a coarse parser for the whole parsing process as well, as long as the coarse parser is efficient. Since supertag disambiguation is highly constrained by the grammar, incorporating grammar constraints into supertagging (including training and testing) by using the CFG-filter can further improve the supertagging performance, as shown in Table 2.

As for the supertagger training time, incorporating grammar constraints inevitably increases the training time. As shown in Table 3, the total training time of forest-guided training (default settings, with chart size limited to 20k) was about 5.8 hours. For each iteration of the FT model, we find that the training time gradually decreases with each successive iteration. This hints that we can do better model initialization to further reduce the training time.

4.4 HPSG Parsing Results

We evaluated the HPSG parsers using different supertagger training methods. For the baseline HPSG parser, a CFG-filter is already incorporated to accelerate the parsing process. In the following experiments, we fed the parser all the possible supertag candidates with the prediction scores generated by the supertaggers. We controlled the upper bound of the chart size in the CFG-filter to make the parser more efficient.

Table 4 shows the results of the different parsing models. We first compared the baseline parsers using different supertaggers. The forest-guided supertagger improved the final FT parser's F-score by 0.68% (statistically significant) over the PW parser using the PW supertagger, which did not consider global grammar constraints during the supertagger training process. The parsing time of the FT parser was very close to that of the PW parser (108s vs. 106s), which was also efficient. The result empirically reflects that incorporating the global grammar constraints into the supertagger training process can refine supertag predicting scores, which become more consistent and compatible with the parser.

We also compared our results with a state-of-the-art HPSG parser using the same grammar. Enju (Miyao, 2005; Ninomiya et al., 2007) is a log-linear model based HPSG parser, which uses a maximum entropy model for the structure disambiguation. In contrast to our baseline parser, full HPSG grammar is directly used with CKY algorithm in the parsing stage. As for the parsing performance, our baseline PW parser using the PW supertagger was 0.23% below the Enju parser. However, by using the forest-guided trained supertagger, our improved FT parser per-

Parser	UP	UR	LP	LR	F-score	Time †
FT Parser	92.28	92.14	89.38	89.23	**89.31**	108s
PW Parser	91.88	91.63	88.75	88.51	88.63	106s
Enju 2.3	92.26	92.21	88.89	88.84	88.86	775s

Table 4: Parser performance on Section 23. "FT Parser" represents baseline parser which uses forest-guided trained supertagger. "PW Parser" represents the baseline parser which uses the point-wise averaged perceptron trained supertagger. (†) The time is the total time of both supertagging and parsing and it was calculated on all 2291 sentences of the Section 23.

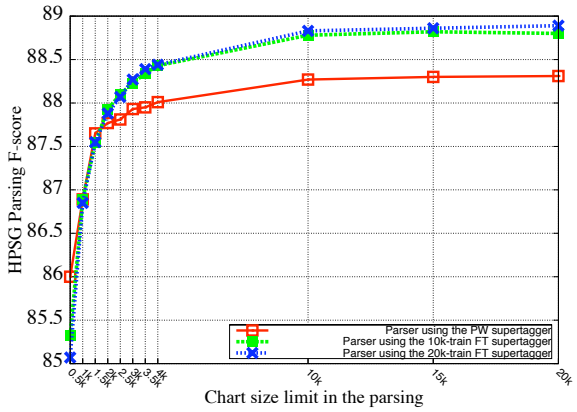


Figure 2: The F-score of the HPSG parsers on section 22 using different settings for the chart size limit in supertagger training and parsing.

formed 0.45% better than the Enju parser (default settings) in F-score. In addition, our shift-reduce style parser was faster than the Enju parser.

Beam size plays an important role for the forest-guided supertagger training method, since a larger beam size reduces the possibility of search errors. Precisely speaking, we control the beam size by limiting the number of edges in the chart in both the forest-guided supertagger training process and the final parsing. Figure 2 shows the results of setting different limits for the chart size during supertagger training and parsing on the development set. The X-axis represents the chart size limitation for the parsing. "10k-train" represents the chart size to be limited to 10k during FT supertagger training phase. A similar representation is used for "20k-train". There is no tree structure search process for the baseline PW supertagger. We evaluated the F-score of the parsers using different supertaggers. As shown in Figure 2, when the chart size of the parser was more than 10k, the benefit of using forest-guided supertaggers were obvious (around an absolute 0.5% improvement in F-score, compared to the parser using the baseline PW supertagger). The performance of the parser using "10k-train" FT supertagger was already approaching to that of the parser using "20k-train" FT supertagger. When the chart size of the parser was less than 2000, the forest-guided supertaggers were not work. Similar to the results showed in previous research (Hal Daumé III and Daniel Marcu, 2005), it is better to use the same chart size limit in the forest-guided supertagger training and the final parsing.

5 Related Work

Since the supertagging technique is well known to drastically improve the parsing speed and accuracy, there is work concerned with tightly integrating a supertagger with a lexicalized grammar parser. Clark and Curran (2004) investigated a multi-tagger supertagging technique for CCG. Based on the multi-tagging technique, supertagger and parser are tightly coupled, in the sense that the parser requests more supertags if it fails. They (Clark and Curran, 2007) also used the perceptron algorithm to train a CCG parser. Different from their work, we focused on improving the performance of the deep parser by refining the training method for supertagging. Ninomiya et al. (2007) used the supertagging probabilities as a reference distribution for the log-linear model for HPSG, which aimed to consistently integrate supertagging into probabilistic HPSG parsing. Prins et al. (2001) trained a POS-tagger on an automatic parser-generated lexical entry corpus as a filter for Dutch HPSG parsing to improve the parsing speed and accuracy.

The existing work most similar to ours is Boullier (2003). He presented a non-statistical parsing-based supertagger for LTAG. Similar to his method, we used a CFG to approximate the original lexicalized grammar. The main difference between these two methods is that we consider the grammar constraints in the training phase of the supertagger, not only in the supertagging test phase and our main objective is to improve the performance of the final parser.

6 Conclusions and Future Work

In this paper, based on the observation that supertaggers are commonly trained separately from lexicalized parsers without global grammar constraints, we proposed a forest-guided supertagger training method to integrate supertagging more tightly with deep parsing. We applied this method to HPSG parsing and made further significant improvement for both supertagging (0.28%) and the HPSG parsing (0.68%) compared to the baseline. The improved parser also achieved a competitive result (89.31%) with a faster parsing speed, compared to a state-of-the-art HPSG parser.

For future work, we will try to weight the forest trees for the supertagger training and extend this method to other lexicalized grammars, such as LTAG and CCG.

Acknowledgments

We are grateful to the anonymous reviewers for their valuable comments. We also thank Goran Topic and Pontus Stenetorp for their help proofreading this paper. The first author was supported by The University of Tokyo Fellowship (UT-Fellowship). This work was partially supported by Grant-in-Aid for Specially Promoted Research (MEXT, Japan).

References

Bangalore, Srinivas and Aravind K. Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25:237–265.

Birch, Alexandra, Miles Osborne, and Philipp Koehn. 2007. CCG supertags in factored statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 9–16.

Boullier, P. 2003. Supertagging: A non-statistical parsing-based approach. In *In Proceedings IWPT-2003*, volume 3, pages 55–65.

Chen, John and Owen Rambow. 2003. Use of deep linguistic features for the recognition and labeling of semantic arguments. In *Proceedings of EMNLP-2003*, pages 41–48.

Clark, Stephen and James R. Curran. 2004. The importance of supertagging for wide-coverage ccg parsing. In *Proceedings of COLING-04*, pages 282–288.

Clark, S. and J.R. Curran. 2007. Perceptron training for a wide-coverage lexicalized-grammar parser. In *Proceedings of the Workshop on Deep Linguistic Processing*, pages 9–16.

Clark, Stephen. 2002. Supertagging for combinatory categorial grammar. In *Proceedings of the 6th International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+ 6)*, pages 19–24.

Collins, M. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP-2002*, pages 1–8.

Hal Daumé III and Daniel Marcu. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *International Conference on Machine Learning (ICML)*, pages 169–176.

Hassan, Hany, Mary Hearne, and Andy Way. 2007. Supertagged phrase-based statistical machine translation. In *Proceedings of ACL-2007*, pages 288–295.

Kiefer, Bernd and Hans-Ulrich Krieger. 2000. A context-free approximation of head-driven phrase structure grammar. In *Proceedings of IWPT-2000*, pages 135–146.

Matsuzaki, Takuya, Yusuke Miyao, and Jun'ichi Tsujii. 2007. Efficient HPSG Parsing with Supertagging and CFG-filtering. In *Proceedings of IJCAI-07*, pages 1671–1676.

Miyao, Yusuke. 2005. Probabilistic disambiguation models for wide-coverage HPSG parsing. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 83–90.

Miyao, Yusuke. 2006. *From Linguistic Theory to Syntactic Analysis: Corpus-Oriented Grammar Development and Feature Forest Model*. Ph.D. Dissertation, The University of Tokyo.

Ninomiya, Takashi, Yoshimasa Tsuruoka, Takuya Matsuzaki, and Yusuke Miyao. 2006. Extremely lexicalized models for accurate and fast HPSG parsing. In *Proceedings of EMNLP-2006*, pages 155–163.

Ninomiya, T., T. Matsuzaki, Y. Miyao, and J. Tsujii. 2007. A log-linear model with an n-gram reference distribution for accurate HPSG parsing. *In Proceedings of IWPT-2007*, pages 60–68.

Pollard, Carl and Ivan A. Sag. 1994. *Head-driven Phrase Structure Grammar*. University of Chicago / CSLI.

Prins, R. and G. Van Noord. 2001. Unsupervised Pos-Tagging Improves Parsing Accuracy And Parsing Efficiency. In *Proceedings of IWPT-2001*, pages 154–165.

Zhang, Yao-zhong, Takuya Matsuzaki, and Jun'ichi Tsujii. 2009. HPSG Supertagging: A Sequence Labeling View. In *Proceedings of IWPT-2009*, pages 210–213.

Entity Linking Leveraging Automatically Generated Annotation

Wei Zhang[†] **Jian Su**[‡] **Chew Lim Tan**[†] **Wen Ting Wang**[‡]

[†]School of Computing
National University of Singapore
{z-wei, tancl}
@comp.nus.edu.sg

[‡] Institute for Infocomm Research
{sujian, wwang}
@i2r.a-star.edu.sg

Abstract

Entity linking refers entity mentions in a document to their representations in a knowledge base (KB). In this paper, we propose to use additional information sources from Wikipedia to find more name variations for entity linking task. In addition, as manually creating a training corpus for entity linking is labor-intensive and costly, we present a novel method to automatically generate a large scale corpus annotation for ambiguous mentions leveraging on their unambiguous synonyms in the document collection. Then, a binary classifier is trained to filter out KB entities that are not similar to current mentions. This classifier not only can effectively reduce the ambiguities to the existing entities in KB, but also be very useful to highlight the new entities to KB for the further population. Furthermore, we also leverage on the Wikipedia documents to provide additional information which is not available in our generated corpus through a domain adaption approach which provides further performance improvements. The experiment results show that our proposed method outperforms the state-of-the-art approaches.

1 Introduction

The named entity (NE) ambiguation has raised serious problems in many areas, including web people search, knowledge base population (KBP), and information extraction, because an entity (such as *Abbott Laboratories*, a diversified pharmaceuticals health care company) can be referred to by multiple mentions (e.g. "*ABT*" and "*Abbott*"), and a mention (e.g. "*Abbott*") can be shared by different entities (e.g. *Abbott Texas*: a city in United States; *Bud Abbott*, an American actor; and *Abbott Laboratories*, a diversified pharmaceutical health care company).

Both Web People Search (WePS) task (Artiles et al. 2007) and Global Entity Detection & Recognition task (GEDR) in Automatic Content Extraction 2008 (ACE08) disambiguate entity mentions by clustering documents with these mentions. Each cluster then represents a unique entity. Recently entity linking has been proposed in this field. However, it is quite different from the previous tasks.

Given a knowledge base, a document collection, entity linking task as defined by KBP-09[1] (McNamee and Dang, 2009) is to determine for each name string and the document it appears, which knowledge base entity is being referred to, or if the entity is a new entity which is not present in the reference KB.

Compared with GEDR and WePS, entity linking has a given entity list (i.e. the reference KB) to which we disambiguate the entity mentions. Moreover, in document collection, there are new entities which are not present in KB and can be used for further population. In fact, new entities with or without the names in KB cover more than half of testing instances.

[1] http://apl.jhu.edu/~paulmac/kbp.html

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1290–1298,
Beijing, August 2010

Entity linking has been explored by several researchers. Without any training data available, most of the previous work ranks the similarity between ambiguous mention and candidate entities through Vector Space Model (VSM). Since they always choose the entity with the highest rank as the answer, the ranking approaches hardly detect a situation where there may be a new entity that is not present in KB. It is also difficult to combine bag of words (BOW) with other features. For example, to capture the "category" information, the method of Cucerzan (2007) involves a complicated optimization issue and the approach has to be simplified for feasible computation, which compromises the accuracy. Besides unsupervised methods, some supervised approaches (Agirre et al. 2009, Li et al. 2009 and McNamee et al. 2009) also have been proposed recently for entity linking. However, the supervised approaches for this problem require large amount of training instances. But manually creating a corpus is labor-intensive and costly.

In this paper, we explore how to solve the entity linking problem. We present a novel method that can automatically generate a large scale corpus for ambiguous mentions leveraging on their unambiguous synonyms in the document collection. A binary classifier based on Support Vector Machine (SVM) is trained to filter out some candidate entities that are not similar to ambiguous mentions. This classifier can effectively reduce the ambiguities to the existing entities in KB, and it is very useful to highlight the new entities to KB for the further population. We also leverage on the Wikipedia documents to provide additional information which is not available in our generated corpus through a domain adaption approach which provides further performance improvements. Besides, more information sources for finding more variations also contribute to the overall 22.9% accuracy improvements on KBP-09 test data over baseline.

The remainder of the paper is organized as follows. Section 2 reviews related work for entity linking. In Section 3 we detail our algorithm including name variation and entity disambiguation. Section 4 describes the experimental setup and results. Finally, Section 5 concludes the paper.

2 Related Work

The crucial component of entity linking is the disambiguation process. Raphael et al. (2007) report a disambiguation algorithm for geography. The algorithm ranks the candidates based on the manually assigned popularity scores in KB. The class with higher popularity will be assigned higher score. It causes that the rank of entities would never change, such as Lancaster (California) would always have a higher rank than Lancaster (UK) for any mentions. However, as the popularity scores for the classes change over time, it is difficult to accurately assign dynamic popularity scores. Cucerzan (2007) proposes a disambiguation approach based on vector space model for linking ambiguous mention in a document with one entity in Wikipedia. The approach ranks the candidates and chooses the entity with maximum agreement between the contextual information extracted from Wikipedia and the context of a document, as well as the agreement among the category tags associated with the candidate entities. Nguyen and Cao (2008) refer the mentions in a document to KIM (Popov et al. 2004) KB. KIM KB is populated with over 40,000 named entities. They represent a mention and candidates as vectors of their contextual noun phrase and co-occurring NEs, and then the similarity is determined by the common terms of the vectors and their associated weights. For linking mentions in news articles with a Wikipedia-derived KB (KBP-09 data set), Varma et al. (2009) rank the entity candidates using a search engine. Han and Zhao (2009) rank the candidates based on BOW and Wikipedia semantic knowledge similarity.

All the related work above rank the candidates based on the similarity between ambiguous mention and candidate entities. However, the ranking approach hardly detects the new entity which is not present in KB.

Some supervised approaches also have been proposed. Li et al. (2009) and McNamee et al. (2009) train their models on a small manually created data set containing only 1,615 examples. But entity linking requires large training data. Agirre et al. (2009) use Wikipedia to construct their training data by utilizing Inter-Wikipedia links and the surrounding snippets of text. However, their training data is created from a

different domain which does not work well in the targeted news article domain.

3 Approach

In this section we describe our two-stage approach for entity linking: name variation and entity disambiguation. The first stage finds variations for every entity in the KB and generates an entity candidate set for a given query. The second stage is entity disambiguation, which links an entity mention with the real world entity it refers to.

3.1 Name Variation

The aim for Name Variation is to build a Knowledge Repository of entities that contains vast amount of world knowledge of entities like name variations, acronyms, confusable names, spelling variations, nick names etc. We use Wikipedia to build our knowledge repository since Wikipedia is the largest encyclopedia in the world and surpasses other knowledge bases in its coverage of concepts and up-to-date content. We obtain useful information from Wikipedia by the tool named Java Wikipedia Library[2] (Zesch et al. 2008), which allows to access all information contained in Wikipedia.

Cucerzan (2007) extracts the name variations of an entity by leveraging four knowledge sources in Wikipedia: "entity pages", "disambiguation pages" "redirect pages" and "anchor text".

Entity page in Wikipedia is uniquely identified by its title – a sequence of words, with the first word always capitalized. The title of Entity Page represents an unambiguous name variation for the entity. A redirect page in Wikipedia is an aid to navigation. When a page in Wikipedia is redirected, it means that those set of pages are referring to the same entity. They often indicate synonym terms, but also can be abbreviations, more scientific or more common terms, frequent misspellings or alternative spellings etc. Disambiguation pages are created only for ambiguous mentions which denote two or more entities in Wikipedia, typically followed by the word "*disambiguation*" and containing a list of references to pages for entities that share the same name. This is more useful in extracting the abbreviations of entities, other possible names for an entity etc. Besides, both outlinks and inlinks in Wikipedia are associated with anchor texts that represent name variations for the entities.

Using these four sources above, we extracted name variations for every entity in KB to form the Knowledge Repository as Cucerzan's (2007) method. For example, the variation set for entity *E0272065* in KB is {*Abbott Laboratories, Abbott Nutrition, Abbott* …}. Finally, we can generate the entity candidate set for a given query using the Knowledge Repository. For example, for the query containing "*Abbott*", the entity candidate set retrieved is {*E0272065, E0064214* ...}.

From our observation, for some queries the retrieved candidate set is empty. If the entity for the query is a new entity, not present in KB, empty candidate set is correct. Otherwise, we fail to identify the mention in the query as a variation, commonly because the mention is a misspelling or infrequently used name. So we propose to use two more sources "Did You Mean" and "Wikipedia Search Engine" when Cucerzan (2007) algorithm returns empty candidate set. Our experiment results show that both proposed knowledge sources are effective for entity linking. This contributes to a performance improvement on the final entity linking accuracy.

Did You Mean: The "*did you* mean" feature of Wikipedia can provide one suggestion for misspellings of entities. This feature can help to correct the misspellings. For example, "Abbot Nutrition" can be corrected to "Abbott Nutrition".

Wikipedia Search Engine: This key word based search engine can return a list of relevant entity pages of Wikipedia. This feature is more useful in extracting infrequently used name.

Algorithm 1 below presents the approach to generate the entity candidate set over the created Knowledge Repository. $\boldsymbol{Ref_E(s)}$ is the entity set indexed by mention ***s*** retrieved from Knowledge Repository. In Step 8, we use the longest common subsequence algorithm to measure the similarity between strings ***s*** and the title of the entity page with highest rank. More details about longest common subsequence algorithm can be found in Cormen et al. (2001).

[2] http://www.ukp.tu-darmstadt.de/software/JWPL

Algorithm 1 Candidate Set Generation

Input: mention *s;*

1: **if** $Ref_E(s)$ is empty
2: *s'* ← Wikipedia"did you mean"Suggestion
3: **If** s' is not NULL
4: s ← *s'*
5: **else**
6: EntityPageList ← WikipediaSearchEngine(s)
7: EntityPage←FirstPage of EntityPageList
8: Sim=Similarity(s,EntityPage.title)
9: **if** Sim > Threshold
10: *s* ← *EntityPage.title*
11: **end if**
12: **end if**
13: **end if**

Output: $Ref_E(s)$;

3.2 Entity Disambiguation

The disambiguation component is to link the mention in query with the entity it refers to in candidate set. If the entity to which the mention refers is a new entity which is not present in KB, *nil* will be returned. In this Section, we will describe the method for automatic data creation, domain adaptation from Wikipedia data, and our supervised learning approach as well.

3.2.1 Automatic Data Creation

The basic idea is to take a document with an unambiguous reference to an entity E1 and replacing it with a phrase which may refer to E1, E2 or others.

Observation: Some full names for the entities in the world are unambiguous. This phenomenon also appears in the given document collection of entity linking. The mention "*Abbott Laboratories*" appearing at multiple locations in the document collection refers to the same entity "*a pharmaceuticals health care company*" in KB.

From this observation, our method takes into account the mentions in the Knowledge Repository associated with only one entity and we treat these mentions as unambiguous name. Let us take *Abbott Laboratories*-{*E0272065*} in the Knowledge Repository as an example. We first use an index and search tool to find the documents with unambiguous mentions. Such as, the mention "*Abbott Laboratories" occurs in* document *LDC2009T13* and *LDC2007T07* in the document collection. The chosen text indexing and searching tool is the well-known Apache Lucene information retrieval open-source library[3].

Next, to validate the consistency of NE type between entities in KB and in document, we run the retrieved documents through a Named Entity Recognizer, to tag the named entities in the documents. Then we link the document to the entity in KB if the document contains a named entity whose name exactly matches with the unambiguous mention and type (i.e. Person, Organization and Geo-Political Entity) exactly matches with the type of entity in KB. In this example, after Named Entity Recognition, "Abbott Laboratories" in document *LDC2009T13* is tagged as an Organization which is consistent with the entity type of *E0272065* in KB. We link the *"Abbott Laboratories"* occurring in *LDC2009T13* with entity *E0272065.*

Finally, we replace the mention in the selected documents with the ambiguous synonyms. For example, we replace the mention *"Abbott Laboratories"* in document *LDC2009T13* with *"Abbott"* where *Abbott-{E0064214, E0272065…}* is an entry in Knowledge Repository. *"Abbott"* is ambiguous, because it is referring not only to *E0272065*, but also to *E0064214* in Knowledge Repository. Then, we can get two instances for the created data set as Figure 1, where one is positive and the other is negative.

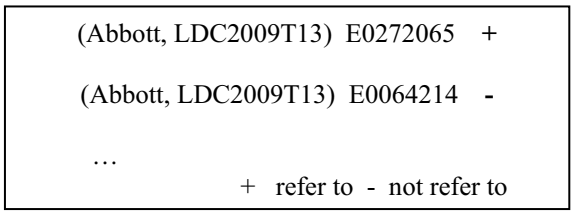


Figure 1: An instance of the data set

However, from our studies, we realize some limitations on our training data. For example, as shown in Figure 1, the negative instance for *E0272065* and the positive instance for

[3] http://lucene.apache.org

E0064214 are not in our created data set. However, those instances exist in the current document collection. We do not retrieve them since there is no unambiguous mention for *E0064214* in the document collection.

To reduce the effect of this problem, we propose to use the Wikipedia data as well, since Wikipedia data has training examples for all the entities in KB. Articles in Wikipedia often contain mentions of entities that already have a corresponding article, and at least the first occurrence of the mentions of an entity in a Wikipedia article must be linked to its corresponding Wikipedia article, if such an article exists. Therefore, if the mention is ambiguous, the hyperlink is disambiguating it. Next, we will describe how to incorporate Wikipedia data.

Incorporating Wikipedia Data. The document collection for entity linking is commonly from other domains, but not Wikipedia. To benefit from Wikipedia data, we introduce a domain adaption approach (Daumé III, 2007) which is suitable for this work since we have enough "target" domain data. The approach is to augment the feature vectors of the instances. Denote by X the input space, and by Y the output space, in this case, X is the space of the real vectors $\varphi(x)$ for the instances in data set and Y= {+1,-1} is the label. D^s is the Wikipedia domain dataset and D^t is our automatically created data set. Suppose for simplicity that $X=R^F$ for some $F > 0$ (R^F is the space of F-dimensions). The augmented input space will be defined by $\breve{X} = R^{3F}$. Then, define mappings $\varphi^s, \varphi^t : X \rightarrow \breve{X}$ for mapping the Wikipedia and our created data set respectively. These are defined as follows:

$$\varphi^s(x) = < \varphi(x), \varphi(x), 0 >$$

$$\varphi^t(x) = < \varphi(x), 0, \varphi(x) >$$

Where $\mathbf{0} = <0,0,\ldots,0> \in R^F$ is the zero vector. We use the simple linear kernel in our experiments. However, the following kernelized version can help us to gain some insight into the method. K denotes the dot product of two vectors. $K(x,x') = < \varphi(x), \varphi(x') >$. When the domain is the same, we get: $\breve{K}(x,x') = < \varphi(x), \varphi(x') > + < \varphi(x), \varphi(x') = 2K(x,x')$. When they are from different domains, we get: $\breve{K}(x,x') = < \varphi(x), \varphi(x') > = K(x,x')$. Putting this together, we have:

$$\breve{K} = \begin{cases} 2K(x,x') & \text{same domain} \\ K(x,x') & \text{diff. domain} \end{cases}$$

This is an intuitively pleasing result. Loosely speaking, this means that data points from our created data set have twice as much influence as Wikipedia points when making predictions about test data from document collection.

3.2.2 The Disambiguation Framework

To disambiguate a mention in document collection, the ranking method is to rank the entities in candidate set based on the similarity score. In our work, we transform the ranking problem into a classification problem: deciding whether a mention refers to an entity on an SVM classifier. If there are 2 or more than 2 candidate entities that are assigned *positive* label by the binary classifier, we will use the baseline system (explained in Section 4.2) to rank the candidates and the entity with the highest rank will be chosen.

In the learning framework, the training or testing instance is formed by (*query*, *entity*) pair. For Wikipedia data, (*query*, *entity*) is *positive* if there is a hyperlink from the article containing the mention in query to the entity, otherwise (*query*, *entity*) is *negative*. Our automatically created data has been assigned labels in Section 3.2.1. Based on the training instances, a binary classifier is generated by using particular learning algorithm. During disambiguation, (*query*, *entity*) is presented to the classifier which then returns a class label.

Each (*query*, *entity*) pair is represented by the feature vector using different features and similarity metrics. We chose the following three classes of features as they represent a wide range of information - lexical features, word-category pair, NE type - that have been proved to be effective in previous works and tasks. We now discuss the three categories of features used in our framework in details.

Lexical features. For Bag of Words feature in Web People Search, Artiles et al. (2009) illustrated that noun phrase and n-grams longer than 2 were not effective in comparison with token-based features and using bi-grams gives the best

results only reaching recall 0.7. Thus, we use token-based features. The similarity metric we choose is cosine (using standard tf.idf weighting). Furthermore, we also take into account the co-occurring NEs and represent it in the form of token-based features. Then, the single cosine similarity feature is based on Co-occurring NEs and Bag of Words.

Word Category Pair. Bunescu (2007) demonstrated that word-category pairs extracted from the document and Wikipedia article are a good signal for disambiguation. Thus we also consider word-category pairs as a feature class, i.e., all *(w,c)* where *w* is a word from Bag of Words of document and *c* is a category to which candidate entity belongs.

NE Type. This feature is a single binary feature to guarantee that the type of entity in document (i.e. Person, Geo-Political Entity and Organization) is consistent with the type of entity in KB.

4 Experiments and Discussions

4.1 Experimental Setup

In our study**,** we use KBP-09 knowledge base and document collection for entity linking. In the current setting of KBP-09 Data, the KB has been generated automatically from Wikipedia. The KB contains 818,741 different entities. The document collection is mainly composed of newswire text from different press agencies. The collection contains 1.3 million documents that span from 1994 to the end of 2008. The test data has 3904 queries across three named entity types: Person, Geo-Political Entity and Organization. Each query contains a document with an ambiguous mention.

Wikipedia data can be obtained easily from the website[4] for free research use. It is available in the form of database dumps that are released periodically. In order to leverage various information mentioned in Section 3.1 to derive name variations, make use of the links in Wikipedia to generate our training corpus and get word category information for the disambiguation, we further get Wikipedia data directly from the website. The version we used in our experiments was released on Sep. 02, 2009. The automatically created corpus (around 10K) was used as the training data, and 30K training instances associated with the entities in our corpus was derived from Wikipedia.

For pre-processing, we perform sentence boundary detection and Chunking derived from Stanford parser (Klein and Manning, 2003), Named Entity Recognition using a SVM based system trained and tested on ACE 2005 with 92.5(P) 84.3(R) 88.2(F), and coreference resolution using a SVM based coreference resolver trained and tested on ACE 2005 with 79.5%(P), 66.7%(R) and 72.5%(F).

We select SVM as the classifier used in this paper since SVM can represent the stat-of-the-art machine learning algorithm. In our implementation, we use the binary SVM^{Light} developed by Joachims (1999). The classifier is trained with default learning parameters.

We adopt the measure used in KBP-09 to evaluate the performance of entity linking. This measure is micro-averaged accuracy: the number of correct link divided by the total number of queries.

4.2 Baseline Systems

We build the baseline using the ranking approach which ranks the candidates based on similarity between mention and candidate entities. The entity with the highest rank is chosen. Bag of words and co-occurring NEs are represented in the form of token-based feature vectors. Then tf.idf is employed to calculate similarity between feature vectors.

To make the baseline system with token-based features state-of-the-art, we conduct a series of experiments. Table 1 lists the performances of our token-based ranking systems. In our experiment, local tokens are text segments generated by a text window centered on the mention. We set the window size to 55, which is the value that was observed to give optimum performance for the disambiguation problem (Gooi and Allan, 2004). Full tokens and NE are all the tokens and named entities co-occurring in the text respectively. We notice that tokens of the full text as well as the co-occurring named entity produce the best baseline performance, which we use for the further experiment.

[4] http://download.wikipedia.org

	Micro-averaged Accuracy
local tokens	60.0
local tokens + NE	60.6
full tokens + NE	61.9

Table 1: Results of the ranking methods

4.3 Experiment and Result

As discussed in Section 3.1, we exploit two more knowledge sources in Wikipedia: "did you mean" (DYM) and "Wikipedia search engine" (SE) for name variation step. We conduct some experiments to compare our name variation method using Algorithm 1 in Section 3.1 with the name variation method of Cucerzan (2007). Table 2 shows the comparison results of different name variation methods for entity linking. The experiments results show that, in entity linking task, our name variation method outperforms the method of Cucerzan (2007) for both entity disambiguation methods.

Name Variation Approaches	Ranking Method	Our Disambiguation Method
Cucerzan (2007)	60.9	82.2
+DYM+SE	61.9	**83.8**

Table 2: Entity Linking Result for two name variation approaches. Column 1 used the baseline method for entity disambiguation step. Column 2 used our proposed entity disambiguation method.

Table 3 compares the performance of different methods for entity linking on the KBP-09 test data. Row 1 is the result for baseline system. Row 2 and Row 3 show the results training on Wikipedia data and our automatically data respectively. Row 4 is the result training on both Wikipedia and our created data using the domain adaptation method mentioned in Section 3.2.1. It shows that our method trained on the automatically generated data alone significantly outperforms baseline. Compared Row 3 with Row 2, our created data set serves better at training the classifier than Wikipedia data. This is due to the reason that Wikipedia is a different domain from newswire domain. By comparing Row 4 with Row 3, we find that by using the domain adaptation method in Section 3.2.1, our method for entity linking can be further improved by 1.5%. Likely, this is because of the limitation of the auto-generated corpus as discussed in Section 3.2.1. In another hand, Wikipedia can complement the missing information with the auto-generated corpus. So combining Wikipedia data with our generated data can achieve better result. Compared with baseline system using Cucerzan (2007) name variation method in Table 2, in total our proposed method achieves a significant 22.9% improvement.

	Micro-averaged Accuracy
Baseline	61.9
Wiki	79.9
Created Data	82.3
Wiki → Created Data	**83.8**

Table 3: Micro-averaged Accuracy for Entity Linking

To test the effectiveness of our method to deal with new entities not present in KB and existing entities in KB respectively, we conduct some experiments to compare with Baseline. Table 4 shows the performances of entity linking systems for existing entities (non-NIL) in KB and new entity (NIL) which is not present in KB. We can see that the binary classifier not only effectively reduces the ambiguities to the existing entities in KB, but also is very useful to highlight the new entities to KB for the further population. Note that, in baseline system, all the new entities are found by the empty candidate set of name variation process, while the disambiguation component has no contribution. However, our approach finds the new entities not only by the empty candidate set, but also leveraging on disambiguation component which also contributes to the performance improvement.

	non-NIL	NIL
Baseline	72.6	52.4
Wiki → Created Data	79.2	87.8

Table 4: Entity Linking on Existing and New Entities

Finally, we also compare our method with the top 5 systems in KBP-09. Among them, *Siel_093* (Varma et al. 2009) and *NLPR_KBP1* (Han and Zhao 2009) use similarity ranking approach; *Stanford_UBC2* (Agirre et al. 2009), *QUANTA1* (Li et al. 2009) and *hltcoe1* (McNamee et al. 2009) use supervised approach. From the results shown in Figure 2, we observe that our method outperforms all the top 5 systems and the baseline system of KBP-09. Specifically, our method achieves better result than both similarity ranking approaches. This is due to the limitations of the ranking approach which have been discussed in Section 2. We also observe that our method gets a 5% improvement over *Stanford_UBC2*. This is because they collect their training data from Wikipedia which is a different domain from document collection of entity linking, news articles in this case; while our automatic data generation method can create a data set from the same domain as the document collection. Our system also outperforms *QUANTA1* and *hltcoe1* because they train their model on a small manually created data set (1,615 examples), while our method can automatically generate a much larger data set.

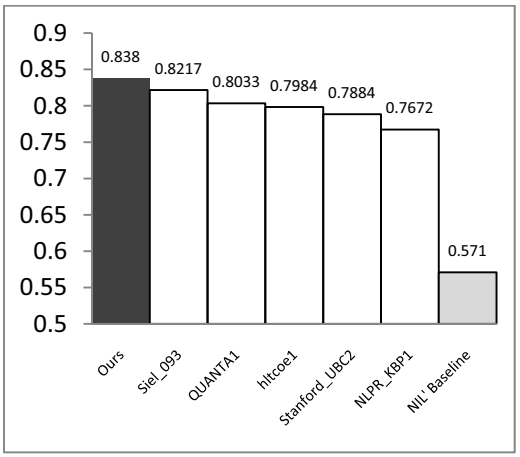


Figure 2: A comparison with KBP09 systems

5 Conclusion

The purpose of this paper is to explore how to leverage the automatically generated large scale annotation for entity linking. Traditionally, without any training data available, the solution is to rank the candidates based on similarity. However, it is difficult for the ranking approach to detect a new entity that is not present in KB, and it is also difficult to combine different features. In this paper, we create a large corpus for entity linking by an automatic method. A binary classifier is then trained to filter out KB entities that are not similar to current mentions. We further leverage on the Wikipedia documents to provide other information which is not available in our generated corpus through a domain adaption approach. Furthermore, new information sources for finding more variations also contribute to the overall 22.9% accuracy improvements on KBP-09 test data over baseline.

References

E. Agirre et al. Stanford-UBC at TAC-KBP. In *Proceedings of Test Analysis Conference 2009 (TAC 09).*

J. Artiles, J. Gonzalo, and S. Sekine. 2007. The semeval-2007 web evaluation: Establishing a benchmark for the web people search task. In *Proceeding of the Fourth International Work-shop on Semantic Evaluations (SemEval-2007).*

J. Artiles, E. Amigo and J. Gonzalo. 2009. The role of named entities in Web People Search. In *proceeding of the 47th Annual Meeting of the Association for Computational Linguistics.*

R. Bunescu. 2007. Learning for information extraction from named entity recognition and disambiguation to relation extraction. Ph.D thesis, University of Texas at Austin, 2007.

T. H. Cormen, et al. 2001. Introduction To Algorithms (Second Edition). *The MIT Press*, Page 350-355.

S. Cucerzan. 2007. Large-Scale Named Entity Disambiguation Based on Wikipedia Data. *Empirical Methods in Natural Language Processing*, June 28-30, 2007.

H. Daumé III. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics* .

C. H. Gooi and J. Allan. 2004. Cross-document coreference on a large scale corpus. In *proceedings of Human Language Technology Conference North American Association for Computational Linguistics Annual Meeting,* Boston, MA.

X. Han and J. Zhao. NLPR_KBP in TAC 2009 KBP Track: A Two-Stage Method to Entity Linking. In *Proceedings of Test Analysis Conference 2009 (TAC 09).*

T. Joachims. 1999. Making large-scale SVM learning practical. In *Advances in Kernel Methods - Support Vector Learning*. MIT Press.

D. Klein and C. D. Manning. 2003. Fast Exact Inference with a Factored Model for Natural Language Parsing. *In Advances in Neural Information Processing Systems 15 (NIPS 2002), Cambridge, MA: MIT Press, pp. 3-10.*

F. LI et al. THU QUANTA at TAC 2009 KBP and RTE Track. In *Proceedings of Test Analysis Conference 2009 (TAC 09).*

P. McNamee and H. T. Dang. 2009. Overview of the TAC 2009 Knowledge Base Population Track. In *Proceedings of Test Analysis Conference 2009 (TAC 09).*

P. McNamee et al. HLTCOE Approaches to Knowledge Base Population at TAC 2009. In *Proceedings of Test Analysis Conference 2009 (TAC 09).*

H. T. Nguyen and T. H. Cao. 2008. Named Entity Disambiguation on an Ontology Enriched by Wikipedia. *2008 IEEE International Conference on Research, Innovation and Vision for the Future in Computing & Communication Technologies.*

B. Popov et al. 2004. KIM - a Semantic Platform for Information Extraction and Retrieval. In *Journal of Natural Language Engineering*, Vol. 10, Issue 3-4, Sep 2004, pp. 375-392, Cambridge University Press.

V. Raphael, K. Joachim and M. Wolfgang, 2007. Towards ontology-based disambiguation of geographical identifiers. In *Proceeding of the 16th WWW workshop on I3: Identity, Identifiers, Identifications, 2007.*

V. Varma et al. 2009. IIIT Hyderabad at TAC 2009. In *Proceedings of Test Analysis Conference 2009 (TAC 09).*

T. Zesch, C. Muller and I. Gurevych. 2008. Extractiong Lexical Semantic Knowledge from Wikipedia and Wiktionary. In *Proceedings of the Conference on Language Resources and Evaluation (LREC)*, 2008.

A Rhetorical Syntax-Driven Model for Speech Summarization

Jian Zhang and Pascale Fung
Human Language Technology Center
Department of Electronic & Computer Engineering
Hong Kong University of Science & Technology (HKUST)
{zjustin, pascale}@ece.ust.hk

Abstract

We show a novel approach of parsing and reordering rhetorical syntax tree for extractive summarization of presentation speech. Our previous work showed (Fung et al., 2008) that rhetorical structures are embedded in this type of speech and that exploring this structure helps improve summarization quality. We further demonstrate that speakers do not follow the strict order of bullet points in the presentation slides, and that a re-ordering of these points occurs. We therefore propose a method of parsing presentation transcriptions into a rhetorical syntax tree and then re-order the leaf nodes to transform the speech transcriptions into an extractive summary, akin to a process of presentation slide generation. Chunking, parsing, and reordering are carried out by 28-class Hidden Markov Support Vector Machine(HMSVM) classifier trained from reference presentations and presentation slides. Using ROUGE-L F-measure we showed that our rhetorical syntax-driven model gives a 35.8% relative improvement over a binary summarizer with no rhetorical information, a 14.3% improvement over Rhetorical State Hidden Markov Model(RSHMM) (Fung et al., 2008), and a 4.3% improvement over our proposed model with no reordering.

1 Introduction

In this paper, we propose to improve extractive summarization of presentation speech using parsing and reordering of the salient points in the speech. Presentation speech includes classroom lectures, conference talks, business seminars, as well as political debates and parliamentary speech where the speaker gives a presentation according to some prepared slides containing bullet points. Some of the speech are transcribed into text, others might even be accompanied by short abstracts. Nevertheless, for learning and collaboration purposes, transcribed text is too long to read whereas short abstracts do not contain enough information (Teufel and Moens, 2002). Especially, in our conference presentation corpus, on average only about 40% bullet points of each transcription appears in the corresponding conference paper abstract. The accompanying presentation slides, on the other hand, are much better in summarizing the gists.

In recent years, more research has been conducted on exploring the hierarchical structure for better summarization performance (Fung et al., 2003; Murray et al., 2006; Sauper and Regina, 2009; Tatar et al., 2008). Unlike text documents, the structure of a spoken document is not immediately apparent in terms of its layout. However, researchers have shown that structural characteristics of a speech are clearly rendered by not just its linguistic features but also its acoustic features (Fung et al., 2008; Hirschberg and Nakatani, 1996; Nakatani et al., 1995). The hierarchical layout structure of Power Point slides enhances

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1299–1307,
Beijing, August 2010

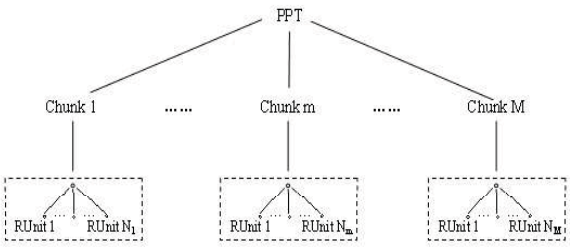


Figure 1: Rhetorical syntax tree

the understanding by the audience. In fact, they even provide a kind of extractive summarization that is superior in terms of informativeness than short abstracts. Unfortunately, presentation slides are not always made available to the audience or for the archive. In some cases, presentation slides consist of mostly figures and graphs, even videos, but without sufficient text bullet points to summarize the content. Meanwhile, there are significant amounts of presentation speech online (e.g. political speech, lectures and seminars) that can be rendered more useful if we can summarize them in a format similar to presentation slides.

Following previous research showing that modeling hierarchical structure indeed helps improve summarization performance, we are interested in going a step further in proposing a rhetorical syntax driven summarization model. Presentation speech is transcribed automatically by an ASR system, then "parsed" into a rhetorical syntax tree. The leaf nodes of the tree, representing actual utterances with rhetorical unit labels, are then "reordered" and organized into a target summary. In this paper, we also propose using a HMSVM (Altun et al., 2003) for the parser and the summarizer. HMSVM has the advantage of considering the interdependence between neighboring sentences. Reordering rules are automatically learned and candidate sequences generated, before they are scored by the final summarizer.

This paper is organized as follows: Section 2 describes our motivation, and the rhetorical structure characteristics in lecture speech. Section 3 details how to parse rhetorical structure of the lecture speech. Section 4 describes the reordering process. Section 5 then describes how to produce extractive summaries. We then describe the corpus, how to create reference summaries, the acoustic/prosodic, linguistic and discourse characteristics of lecture speech, baselines, and our in-house automatic speech recognition system are presented in Section 6. Section 7 presents the experiment results. Section 8 describes related work. We then conclude at the end of this paper.

2 Rhetorical Syntax Tree of Presentation Speech

Unlike conversational speech, lectures and presentations are planned. Lecture speakers follow a relatively rigid rhetorical structure at the document level: s/he starts with an overview of the topic to be presented, followed with the actual content with more detailed descriptions, and then concludes at the end. The speech is given in several coherent "chunks" corresponding to the talk outline. By looking at presentation slides as shown in Figure 1, we can clearly see the chunk boundaries, which always exist at slide transitions, delineate content changes. Each of the chunks, in turn, contains many coherent text spans, namely the rhetorical units. Each rhetorical unit contains one or more slides. We represent the rhetorical structure of presentations by a hierarchical text plan, or a rhetorical tree. Since lecture speeches are mostly based on presentation slides with main bullet points, the structural format of the presentation slides is a faithful representation of the document-level rhetorical structure of the speech. At the top level of the tree are the rhetorical chunks, and at the lower level child nodes are rhetorical units. Each of the chunks contains several rhetorical units, where each unit may contain a number of utterances corresponding to a list of bullet points in the slides.

We propose using the annotation labels commonly shared by most presentation speech for labeling rhetorical chunks as shown in the left column of Table 1. The chunk label definitions are derived from the general structure of presentations in the specific domain of our corpus, namely conference presentations. Note that whereas we chose to use 7 labels for presentation speech, label definitions are fairly obvious and easy to derive for other genres of speech. We use a machine-aided manual annotation method to label the training presentation speech data. Referring to the slides

Table 1: Rhetorical Chunk and Unit Description

Rhetorical Chunk	Rhetorical Unit
c_1 (Title)	title, author of the presentation
c_2 (Outline)	texture structure;
c_3 (Motivation)	aim; problem/phenomenon
c_4 (Related work)	rival/contrast; continuation
c_5 (Methodology)	solution/inventive step
c_6 (Experiment)	corpus description; detailed experimental setup
c_7 (Conclusion)	conclusion; future work

of each presentation, each sentence in the speech transcription is assigned a label corresponding to one of the 7 chunks defined in Table 1. First, all bullet point sentences in the slides are assigned a chunk label according to its section. Referring to our previous work, a Relaxed Dynamic Time Warping program (Zhang et al., 2008) is used to roughly align transcribed sentences to the corresponding slide bullet points. Chunk labels are also included in this alignment. Human inspection quickly corrects any alignment mistakes made by the program. We then label all the sentence of each type "c_i" rhetorical chunk as "c_i". For example, the sentences of a rhetorical chunk which describes "Methodology: solution/inventive step" is labeled as "c_5".

Rhetorical units are described in the right column of Table 1. The rhetorical units, as we explain below, are clustered automatically without explicit labels. These rhetorical units correspond more or less to the definitions in Table 1, without human effort. To obtain the reference rhetorical unit labels of each type of rhetorical chunks in the entire experiment corpus described in Section 6.1, we cluster all utterances that belong to the same chunk in all presentation speech into several rhetorical units by using modified k-means (MKM) clustering algorithm (Wilpon and Rabiner, 1984; Fung et al., 2003). MKM starts from the centroid of all utterances in one rhetorical chunk and splits the clusters top down until the sub-clusters stabilize. Each final cluster represents one rhetorical unit. The clustering algorithm is shown as follows.

Given all utterances within the same type of chunk from all presentation speech:
(1) **Compute** the centroid;
(2) **Assign** sentence feature vectors closest to each centroid to its cluster;
(3) **Update** each centroid feature vector using all sentence feature vectors assigned to each cluster;
(4) **Iterate** step(2) to step (4) until sentence feature vectors stop moving between clusters;
(5) **Stop** if clusters stabilizes, and **get** final clusters, else **goto** step (6);
(6) **Split** the cluster with largest intra-cluster distance into two by finding the pair of vectors as new centroids, and **repeat** steps (2) to step (5).

Using the above algorithm, we find out the following rhetorical unit clusterings of different kinds of rhetorical chunks on our experiment corpus: (1) two rhetorical units in "Title"(c_1) chunk: "r_i:c_1, $i = 1, 2$";
(2) three rhetorical units in "Outline"(c_2) chunk and "Conclusion"(c_7) chunk:
"r_i:c_j, $i = 1, 2, 3; j = 2, 7$";
(3) five rhetorical units in "Motivation"(c_3) chunk,
"Related work"(c_4) chunk,
"Methodology"(c_5) chunk,
and "Experiment"(c_6) chunk:
"r_i:c_j, $i = 1, 2, 3, 4, 5; j = 3, 4, 5, 6$".

These rhetorical unit clusters correspond roughly to the definitions in the right hand column of Table 1, though no manual labeling is involved.

3 Parsing Presentation Speech

By using our rhetorical syntax-driven model, the process of parsing presentation speech can be described as follows. First we extract acoustic/prosodic features from presentation speech, and linguistic, discourse features from the ASR transcribed text. These features are described in Section 6.1. Next we parse the presentation speech into rhetorical units. Given the ASR transcription of a presentation speech, our task is to parse the transcription sentences into chunks and then into rhetorical units (leaf nodes) that roughly correspond to the rhetorical chunks and units in Table 1 according to their feature vectors. We consider the parsing process as a multi-class classification problem. Each rhetorical unit of each

rhetorical chunk is represented by one class.

Considering that HMSVM (Altun et al., 2003) combines the advantages of maximum margin classifier and kernels with the elegance and efficiency of HMMs, and can effectively handle the dependency between neighboring sentences, we train a twenty-eight-class HMSVM classifier for parsing speech, with one class representing each rhetorical unit. As an example, the sentences labeled as "r_2:c_5" belong to the second rhetorical unit of "Methodology"(c_5) chunk. We have found that, by looking at our corpus of conference presentations, speakers indeed follow the "chunk order" of the slides they use. We add some constraints existing between "r_m:c_i" and "r_n:c_j" where "$i \neq j$" according to the "chunk order" of the slides. Function f_1 maps each given speech or transcription $\mathbf{y}$ to a rhetorical unit label sequence $\mathbf{z}$. For example we want to learn a discriminant function $F_1 : \mathcal{Y} \times \mathcal{Z} \rightarrow \mathcal{R}$ over input/output pairs from which we produce a prediction by maximizing F_1 over the output variable for a given input $\mathbf{y}$. The general form of our hypotheses f_1 is:

$$\mathbf{z}^* = f_1(\mathbf{y}; \mathbf{w}) = \underset{\mathbf{z} \in \mathcal{Z}}{\operatorname{argmax}} F_1(\mathbf{y}, \mathbf{z}; \mathbf{w}) \quad (1)$$

where $\mathbf{w}$ denotes a weighting parameter vector to learn.

We assume F_1 to be linear in some combined feature representation of inputs, described in Section 6.1, and outputs $\Psi(\mathbf{y}, \mathbf{z})$. We then get $F_1(\mathbf{y}, \mathbf{z}; \mathbf{w}) = \langle \mathbf{w}, \Psi(\mathbf{y}, \mathbf{z}) \rangle$. Moreover, we apply a kernel function K over the joint input/output space such that:

$$K((\mathbf{y}, \mathbf{z}), (\overline{\mathbf{y}}, \overline{\mathbf{z}})) = \langle \Psi(\mathbf{y}, \mathbf{z}), \Psi(\overline{\mathbf{y}}, \overline{\mathbf{z}}) \rangle \quad (2)$$

Ψ can be written as a sum over the length of the sequence and decomposed as:

$$\Psi(\mathbf{O}, \mathbf{z}) = (\sum_{i=1}^{l(\mathbf{O})} \Psi_{\sigma,\tau}(\upsilon_i, \upsilon_{i+1}, \mathbf{O}, i))_{\sigma,\tau \in \gamma} \quad (3)$$

where γ is the rhetorical unit label set. $l(\mathbf{O})$ is the length of the observation sequence $\mathbf{O}$ in our case. Ψ is composed by mapping functions that depend only on labels at position i and $i+1$, $\mathbf{O}$ as well as i (Markov property).

We then rewrite F_1 using $\mathbf{w} = (\mathbf{w}_{\sigma,\tau})_{\sigma,\tau \in \gamma}$ as Equation 4.

$$\begin{aligned} F_1(\mathbf{O}, \mathbf{z}) &= \sum_{\sigma,\tau \in \gamma} \langle \mathbf{w}_{\sigma,\tau}, \sum_{i=1}^{l(\mathbf{O})} \Psi_{\sigma,\tau}(\upsilon_i, \upsilon_{i+1}, \mathbf{O}, i) \rangle \\ &= \sum_{i=1}^{l(\mathbf{O})} \underbrace{\sum_{\sigma,\tau \in \gamma} \langle \mathbf{w}_{\sigma,\tau}, \Psi_{\sigma,\tau}(\upsilon_i, \upsilon_{i+1}, \mathbf{O}, i) \rangle}_{=: g(\upsilon_i, \upsilon_{i+1}, \mathbf{O}, i)} \end{aligned} \quad (4)$$

In decoding process, using this decomposition (Altun et al., 2003) we can define

$$\begin{aligned} V(i, \upsilon) := \max_{\upsilon' \in \gamma} (V(i-1, \upsilon')) + g(\upsilon', \upsilon, \mathbf{O}, i-1) \\ when\ i > 1 \\ or\ := 0 \quad otherwise \end{aligned} \quad (5)$$

as the maximal score for all labels with label υ at position i. Using dynamic programming we compute $\max_{\upsilon \in \gamma} V(l(\mathbf{O}), \upsilon)$. The optimal label sequence is recovered by backtracking.

4 Reordering Rhetorical Unit Sequence

4.1 Extracting Reordering Rules

We found that, by looking at our corpus of conference presentations described in Section 6.1, presentation speakers do not always follow the bullet point order within a chunk. When demonstrating a current slide they may be already introducing the next slide. About 11% of the rhetorical units in the transcriptions are out of order vis-a-vis the corresponding bullet points in the presentation slide. As an integral part of our rhetorical syntax-driven summarization model, the rhetorical unit sequence and consequently the sentence sequence are reordered within a chunk. The extraction of reordering rules is based on the alignment between source rhetorical unit sequence in the speech transcription and target rhetorical unit sequence in the Power Point slide sentences. Each sentence is represented by its rhetorical unit label. For example, from the training set and development set in one of our held-out experiment settings described in Section 7, we extracted the following reordering rules: (1)$(r_3, r_1) \rightarrow (r_1, r_3)$;

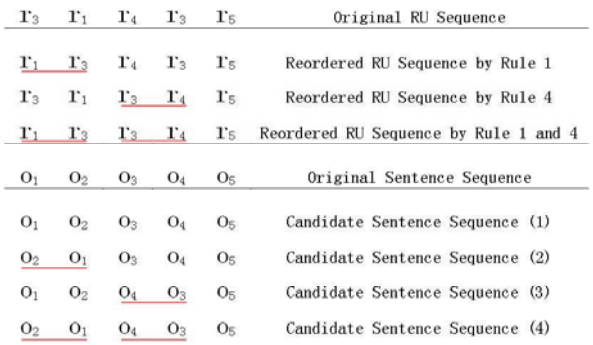


Figure 2: Candidate sentence sequences after applying reordering rules

$(2)(r_3, r_2) \rightarrow (r_2, r_3)$; $(3)(r_4, r_2) \rightarrow (r_2, r_4)$; $(4)(r_4, r_3) \rightarrow (r_3, r_4)$; $(5)(r_5, r_3) \rightarrow (r_3, r_5)$; $(6)(r_5, r_4) \rightarrow (r_4, r_5)$.

In each reordering rule, the left item represents rhetorical unit sequence of the transcription sentences while the right item represents rhetorical unit sequence of bullet points of the corresponding Power Point slides. From these reordering rules, we can see that the speakers in our corpus talk about content described by future bullet points (i.e in the subsequent rhetorical units), but never seem to repeat content from bullet points in the previous unit(s).

4.2 Applying Reordering Rules

Given a sentence sequence and its corresponding rhetorical unit sequence within each chunk, from left to right, with a shifting window of two, we search for the matching reordering rule and adjust the order of the sentences one matched rule at a time, yielding a set of at most 2^L sentence sequence candidates for each chunk where L equals to the length of the sentence. From our data, we found that there are at most 2 matched rules per sentence sequence. So including the original sequence, at most 4 candidate sequences are generated for each chunk.

A reordering example is shown in Figure 2. We apply the reordering rules on a sentence sequence "$(o_1, o_2, o_3, o_4, o_5)$" and the corresponding unit sequence "$(r_3, r_1, r_4, r_3, r_5)$". Four candidate reordered sentence sequence are produced. Without any reordering, we get "Candidate Sentence Sequence (1)". Using reordering Rule 1, we get "Candidate Sentence Sequence (2)".

5 Rhetorical Syntax-driven Summarization

Following sentence reordering, the extractive summarizer selects salient sentences from each chunk using a binary-class classifier. The classifier is run over all candidate sequences from Section 4 and the system selects the best sequence and its summary sentences according to the output probability of the classifier. The best sequence $\{o_1...o_i...o_k\}$ satisfies

$$\text{argmax} \sum_{i=1}^{k} \lg P(o_i \in summary\ sentence\ set | c_j) \quad (6)$$

where c_j represents the rhetorical chunk c_j which has several candidate sequences, including the sequence $\{o_1...o_i...o_k\}$.
$P(o_i \in summary\ sentence\ set|c_j)$ is output probability of that the sentence o_i in the rhetorical chunk c_j is summary sentence.

Again, an HMSVM classifier is used at this stage. The sentence feature vector $\mathbf{o}$ now has its rhetorical unit label as an additional feature, to yield a new sentence feature vector $\hat{\mathbf{o}}$. For the sentence vector sequence $\mathbf{z}$ of each chunk, we label it by using the optimal function $F_2(\mathcal{Z}, \mathcal{V})$. The training stage is similar to that of training the HMSVM parser. The difference is that the HMSVMs for summarization are binary classifiers, while the HMSVM parser is a multi-class classifier.

6 Experimental Setup

6.1 Corpus

We use a lecture speech corpus containing wave files of 71 presentations recorded from different mandarin speakers at two technical conferences, together with well-formatted Power Point slides, manual transcriptions, and their associated audio data. Each presentation lasts about 15 minutes on average. The 71 presentations are split into 391 chunks, and each sentence is assigned a rhetorical chunk label, using the machine-aided human labeling method as described in Section 2. Each

chunk has on average 4.3 rhetorical units. The reference rhetorical unit labels are created by using unsupervised MKM algorithm described in Section 2. Since the labeling process also yields an alignment path between transcription sentences and Power Point slide bullet points, we extract those sentences that have the highest alignment scores with the bullet points to form reference summary sentences, then corrected by five human subjects according to the rhetorical chunk descriptions in the right column of Table 1. We use the kappa coefficient (Krippendorff, 1980) for measuring stability of each annotator and reproducibility between each pair of annotators. The average kappa coefficient is higher than 0.85.

6.2 Features and Baselines

We use the discourse feature PossionNoun proposed in our previous work (Zhang et al., 2008) which is based on the following assumptions: first, if a sentence contains new noun words, it probably contains new information. The noun word's Poisson score varies according to its position. We use Poisson distribution to approximate the variation. Second, if a noun word occurs frequently, it is likely to be more important than other noun words, and the sentence with these high frequency noun words should be included in a summary. We also use the following acoustic and linguistic features for representing sentences. The acoustic features are: duration of the sentence, average syllable duration of the sentence, F0 and Energy min/max/mean/slope/range value of the sentence. The linguistic features are: sentence word count, TF/IDF of each word in the sentence, and the word identity in each sentence.

We use three alternate summarization models for comparison. One is a binary classifier without any rhetorical information, one class for summary sentence and the other for non-summary sentence. The second is RSHMM (Fung et al., 2008). The third is our rhetorical syntax tree without reordering. The two above are built by using acoustic, linguistic, and discourse features for representing the sentences. We apply rhetorical unit label as an additional feature for building our proposed models(with/without reordering).

Table 2: Summarization average performance of 6-fold cross validation experiment in ROUGE-L F-measure (F-measure)

Bianry	RSHMM	Without reordering	Syntax tree with reordering
.53(.52)	.63(.56)	.69(.61)	.72(.65)

(1)Binary: binary classifier as baseline
(2)RSHMM: Rhetorical State HMM proposed in our previous work (Fung et al., 2008)
(3)without reordering: rhetorical syntax tree based summarizer without reordering
(4)Syntax tree with reordering: rhetorical syntax tree based summarizer with reordering

6.3 Lecture Speech ASR Transcription System

We apply our rhetorical syntax-driven summarization model for ASR transcriptions. The database for building our in-house ASR system contains 29 hours of audio data from the technical conferences in our corpus. We choose approximately 21 hours of speech as the training data. The test data comprises of 12 presentations with approximately 3 hours of audio data. Our decoding system runs in multiple passes. Automatic segmentation is first performed on the lecture speech audio data. This is followed by bigram decoding with the gender independent (GI) acoustic model (AM) and lattice generation. Trigram and four-gram branches are created for AM adaptation through lattice expansion and rescoring. Re-decoding with both adapted AM and adapted language model (LM) is performed to produce 1-best results. System combination via recognizer output voting error reduction scheme (ROVER) (Fiscus, 1997) is employed by using character based alignment from the trigram and four-gram branch outputs. The final system obtains a recognition performance of 79.2% character accuracy.

7 Experimental Results

For evaluating different summarization systems, we perform 6-fold cross validation experiments, and two held-out experiments. There are many kinds of metrics for evaluating speech summarization performance (Zhu and Penn, 2005; Penn and Zhu, 2008). We choose ROUGE-L F-

Table 3: Summarization performance of held-out experiments in ROUGE-L F-measure (F-measure)

(A) on manual transcription

Binary	RSHMM	Without reordering	Syntax tree with reordering
.50(.48)	.61(.52)	.67(.59)	.69(.61)

(B) on ASR transcription

Binary	RSHMM	Without reordering	Syntax tree with reordering
.46(.43)	.59(.50)	.66(.57)	.68(.61)

(1)Binary: binary classifier as baseline
(2)RSHMM: Rhetorical State HMM proposed in our previous work (Fung et al., 2008)
(3)without reordering: rhetorical syntax tree based summarizer without reordering
(4)Syntax tree with reordering: rhetorical syntax tree based summarizer with reordering

measure (Lin, 2004) and F-measure (Van Rijsbergen, 1979) as evaluation metrics in our experiments. 11 documents from the 71 presentations are excluded as our development set. In the 6-fold cross validation experiments, we divide the remaining 60 presentations into six subsets of equal size. For each experiment, we use five subsets to train all models and the remaining subset for testing. The average performance of these 6-fold cross validation experiments is shown in Table 2.

Among these 60 presentations, 50 are randomly selected as training data for the two held-out experiments, while the remaining ten are used as test data. Table 3-(A) shows the result of the one held-out experiment on manual transcriptions of test data. Table 3-(B) shows the other held-out experiment on ASR transcriptions of test data.

From these results, we can see that the proposed rhetorical syntax-driven summarizer with reordering outperforms all other methods. Table 2 shows that our rhetorical syntax-driven model gives a 35.8% relative improvement over a binary summarizer with no rhetorical information, a 14.3% improvement over RSHMM (Fung et al., 2008), and a 4.3% improvement over our proposed model with no reordering. These findings suggest that our rhetorical syntax-driven summarization model built by using binary HMSVM classifier apply the sequence information of the sentences within the same rhetorical chunk for improving summarization performance because of the Markov property of HMSVM.

In the above experiments, we all use the rhetorical unit labels produced by our rhetorical structure parser for improving summarization performance. The average accuracy of the rhetorical structure parser is about 83.2%. When we use the reference rhetorical unit labels on our cross-validation experiments, the average summarization performance is 0.75 of ROUGE-L F-measure, a 4.2% improvement over that using the rhetorical unit labels produced by the rhetorical structure parser.

Although the overall performance on ASR transcriptions is worse than that of manual transcriptions, the performance is also satisfying. Furthermore, we also find that using only acoustic features our model obtains satisfying result, 0.65 of ROUGE-L F-measure, on 6-fold cross validation experiment.

8 Related Work

(Furui et al., 2008) has shown that feature-based extractive summarization is an approach that is efficient and more effective than MMR-based approach for lecture speech summarization.

(Marcu, 1997) described the first experiment that shows the concepts of rhetorical analysis and nuclearity can be used effectively for text summarization. (Fung et al., 2003) presented a stochastic HMM framework with modified K-means and segmental K-means algorithms for extractive text summarization. (Fung and Ngai, 2006) further presented a stochastic Hidden Markov Story Model for multilingual and multi-document summarization and proposed that monolingual documents recounting the same story (i.e., in the same topic) share a unique story flow (one story, one flow), and such a flow can be modeled by HMMs. (Barzilay and Lee, 2004) presented an unsupervised method for the induction of content models, which capture constraints on topic selection and organization for texts in a particular domain (Branavan et al., 2007) proposed a structured discriminative model for table-of-contents

generation on written text that accounts for a wide range of phrase-based and collocation features. (Eisenstein and Barzilay, 2008) describes a novel Bayesian approach to unsupervised lexical cohesion driven topic segmentation.

Many researchers have suggested that rhetorical information also exists in spoken documents and efficient modeling of this information is helpful to the summarization task. (Tatar et al., 2008) and (AKITA and Kawahara, 2007) used the Hearst method (Hearst, 1997) to segment documents and detect topics for text summarization and topic adaptation of speech recognition systems for long speech archives respectively. (Hirohata et al., 2005) consider that humans tend to summarize presentations by extracting important sentences from introduction and conclusion sections, and further propose a summarization method based on this structural characteristic. They estimated the introduction and conclusion section boundaries based on the Hearst method (Hearst, 1997), using sentence cohesiveness which is measured by a cosine value between content word-frequency vectors, before performing summarization. Teufel and Moens (Teufel and Moens, 2002) also proposed "rhetorical status" as summary unit, but without hierarchical structure or reordering. Furthermore, many linguists believe that speech acoustics contribute to rhetorical and discourse structure. (Nakatani et al., 1995) provide empirical evidence that discourses can be segmented reliably, and that acoustic characteristics are used by speakers to convey linguistic structure at the discourse level in the English domain. There is a large amount of previous work seeking to demonstrate that acoustic prosodic profile of speech closely models its discourse or rhetorical structure (Halliday, 1967; Ladd, 1996; Hirschberg and Nakatani, 1996).

Our work described in this paper is closely related to (Marcu, 1997; Teufel and Moens, 2002) in that we propose using rhetorical units for summarization. Our method differs from (Teufel and Moens, 2002) in that we assume the relevance or saliency and function of certain text pieces can be determined by analyzing the full hierarchical structure of the text. Instead of annotating the training data with rhetorical labels manually, we propose using Power Point slides as references. (He et al., 2000; He et al., 1999) also investigate the correlation between Power Point slides and extractive summaries. Our learning method is based on classifiers, while (Marcu, 1997) uses rule-based method for parsing rhetorical structure. We train our rhetorical parser by using those reference rhetorical units of the transcriptions created by aligning Power Point slides. We propose a syntax-driven parsing model with reordering for summarization, and we propose a different classifier, HMSVM, for handling the dependency between neighboring sentences within each chunk, when we accomplish our summarization task.

9 Conclusion and Discussion

In this paper, we have shown a rhetorical syntax-driven summarization method for presentation speech. In view of the fact that rhetorical structure in speech is inherently hierarchical, our method chunks, parses, and reorders the presentation utterance sentences before selecting some of them as summary sentences.

We have proposed to use HMSVM classifiers for the parser and the summarizer, taking into account the dependency between neighboring chunks, rhetorical units and sentences with Markov property. Our rhetorical syntax-driven summarizer with reordering outperforms a binary summarizer without rhetorical information with 35.8% relative improvement and outperforms RSHMM (Fung et al., 2008) with 14.3% relative improvement. It gives a 4.3% relative improvement over the same model without reordering. For future work, we are interested in investigating how to apply our rhetorical syntax-driven method to other genres of speech, such as meetings and parliamentary speech.

10 ACKNOWLEDGEMENT

This work is partially supported by ITS/189/09 of the Hong Kong Innovation and Technology Fund(ITF). The authors would like to thank Chan Ho Yin for his valuable input to this work.

References

AKITA, Y., Nemoto Y. and T. Kawahara. 2007. PLSA-based topic detection in meetings for adaptation of lexicon and language model. *Proc. Interspeech 2007*, pages 602–605.

Altun, Y., I. Tsochantaridis, and T. Hofmann. 2003. Hidden Markov Support Vector Machines. In *Machine Learning-International Workshop Then Conference-*, volume 20.

Barzilay, R. and L. Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. *Proceedings of HLT-NAACL*, pages 113–120.

Branavan, SRK, P. Deshpande, and R. Barzilay. 2007. Generating a table-of-contents. In *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, volume 45, page 544.

Eisenstein, J. and R. Barzilay. 2008. Bayesian unsupervised topic segmentation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 334–343. Association for Computational Linguistics.

Fiscus, J.G. 1997. A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (ROVER. In *In Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding*.

Fung, P. and G. Ngai. 2006. One story, one flow: Hidden Markov Story Models for multilingual multidocument summarization. *ACM Transactions on Speech and Language Processing (TSLP)*, 3(2):1–16.

Fung, P., G. Ngai, and P. Cheung. 2003. Combining optimal clustering and hidden Markov models for extractive summarization. *Proceedings of ACL Workshop on Multilingual Summarization*, pages 29–36.

Fung, P., H.Y. Chan, and J.J. Zhang. 2008. Rhetorical-State Hidden Markov Models For Extractive Speech Summarization. *ICASSP2008. Proceedings*, pages 4957–4960.

Furui, Y., K. Yamamoto, N. Kitaoka, and S. Nakagawa. 2008. Class Lecture Summarization Taking into Account Consecutiveness of Important Sentences. *Proceedings of Interspeech 2008*, pages 2438–2441.

Halliday, M.A.K. 1967. *Intonation and grammar in British English*. Mouton.

He, L., E. Sanocki, A. Gupta, and J. Grudin. 1999. Auto-summarization of audio-video presentations. In *Proceedings of the seventh ACM international conference on Multimedia (Part 1)*, page 498. ACM.

He, L., E. Sanocki, A. Gupta, and J. Grudin. 2000. Comparing presentation summaries: slides vs. reading vs. listening. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 177–184. ACM New York, NY, USA.

Hearst, M.A. 1997. TextTiling: Segmenting Text into Multi-paragraph Subtopic Passages. *Computational Linguistics*, 23(1):33–64.

Hirohata, M., Y. Shinnaka, K. Iwano, and S. Furui. 2005. Sentence extraction-based presentation summarization techniques and evaluation metrics. *Proc. ICASSP2005*, 1.

Hirschberg, J. and C.H. Nakatani. 1996. A prosodic analysis of discourse segments in direction-giving monologues. *Proceedings of the 34th conference on Association for Computational Linguistics*, pages 286–293.

Krippendorff, K. 1980. Content analysis: An introduction to its methodology. *Beverly Hills,: Sage Publications*.

Ladd, D.R. 1996. *Intonational Phonology*. Cambridge University Press.

Lin, C.Y. 2004. Rouge: A Package for Automatic Evaluation of Summaries. *Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004)*, pages 25–26.

Marcu, D. 1997. From discourse structures to text summaries. In *Proceedings of the ACL Workshop on Intelligent Scalable Text Summarization*, pages 82–88.

Murray, G., M. Taboada, and S. Renals. 2006. Prosodic correlates of rhetorical relations. In *Proceedings of the Analyzing Conversations in Text and Speech (ACTS) Workshop at HLT-NAACL*, pages 1–7.

Nakatani, C.H., J. Hirschberg, and B.J. Grosz. 1995. Discourse structure in spoken language: Studies on speech corpora. *AAAI 1995 Spring Symposium Series: Empirical Methods in Discourse Interpretation and Generation*, pages 106–112.

Penn, G. and X. Zhu. 2008. A critical reassessment of evaluation baselines for speech summarization. *Proceedings of ACL-HLT. Columbus, OH*.

Sauper, C. and B. Regina. 2009. Automatically Generating Wikipedia Articles: A Structure-Aware Approach. In *Proceedings of the ACL 2009*, pages 208–216.

Tatar, D., E. Tamaianu-Morita, A. Mihis, and D. Lupsa. 2008. Summarization by Logic Segmentation and Text Entailment. *Advances in Natural Language Processing and Applications*, pages 15–26.

Teufel, S. and M. Moens. 2002. Summarizing scientific articles: experiments with relevance and rhetorical status. *Computational Linguistics*, 28(4):409–445.

Van Rijsbergen, CJ. 1979. Information Retrieval. *Buttersworth, London*.

Wilpon, JG and LR Rabiner. 1984. A modified K-means clustering algorithm for use in speaker-independent isolated word recognition. *The Journal of the Acoustical Society of America*, 75:S93.

Zhang, J.J., S. Huang, and P. Fung. 2008. RSHMM++ for extractive lecture speech summarization. In *IEEE Spoken Language Technology Workshop, 2008. SLT 2008*, pages 161–164.

Zhu, X. and G. Penn. 2005. Evaluation of sentence selection for speech summarization. In *ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization*. Citeseer.

Maximum Metric Score Training for Coreference Resolution

Shanheng Zhao and **Hwee Tou Ng**
Department of Computer Science
National University of Singapore
{zhaosh,nght}@comp.nus.edu.sg

Abstract

A large body of prior research on coreference resolution recasts the problem as a two-class classification problem. However, standard supervised machine learning algorithms that minimize classification errors on the training instances do not always lead to maximizing the F-measure of the chosen evaluation metric for coreference resolution. In this paper, we propose a novel approach comprising the use of instance weighting and beam search to maximize the evaluation metric score on the training corpus during training. Experimental results show that this approach achieves significant improvement over the state-of-the-art. We report results on standard benchmark corpora (two MUC corpora and three ACE corpora), when evaluated using the link-based MUC metric and the mention-based B-CUBED metric.

1 Introduction

Coreference resolution refers to the process of determining whether two or more noun phrases (NPs) in a text refer to the same entity. Successful coreference resolution benefits many natural language processing tasks. In the literature, most prior work on coreference resolution recasts the problem as a two-class classification problem. Machine learning-based classifiers are applied to determine whether a candidate anaphor and a potential antecedent are coreferential (Soon et al., 2001; Ng and Cardie, 2002b).

A large body of prior research on coreference resolution follows the same process: during training, they apply standard supervised machine learning algorithms to minimize the number of misclassified training instances; during testing, they maximize either the local or the global probability of the coreferential relation assignments according to the specific chosen resolution method.

However, minimizing the number of misclassified training instances during training does not guarantee maximizing the F-measure of the chosen evaluation metric for coreference resolution. First of all, coreference is a rare relation. There are far fewer positive training instances than negative ones. Simply minimizing the number of misclassified training instances is suboptimal and favors negative training instances. Secondly, evaluation metrics for coreference resolution are based on global assignments. Not all errors have the same impact on the metric score. Furthermore, the extracted training instances are not equally easy to be classified.

In this paper, we propose a novel approach comprising the use of instance weighting and beam search to address the above issues. Our proposed maximum metric score training (MMST) approach performs maximization of the chosen evaluation metric score on the training corpus during training. It iteratively assigns higher weights to the hard-to-classify training instances. The output of training is a standard classifier. Hence, during testing, MMST is faster than approaches which optimize the assignment of coreferential relations during testing. Experimental results show that MMST achieves significant improvements over the baselines. Unlike most of the previous work, we report improved results over the state-of-the-art on all five standard benchmark corpora

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1308–1316,
Beijing, August 2010

(two MUC corpora and three ACE corpora), with both the link-based MUC metric and the mention-based B-CUBED metric.

The rest of this paper is organized as follows. We first review the related work and the evaluation metrics for coreference resolution in Section 2 and 3, respectively. Section 4 describes the proposed MMST algorithm. Experimental results and related discussions are given in Section 5. Finally, we conclude in Section 6.

2 Related Work

Soon *et al.* (2001) proposed a training and testing framework for coreference resolution. During training, a positive training instance is formed by a pair of markables, i.e., the anaphor (a noun phrase) and its closest antecedent (another noun phrase). Each markable (noun phrase) between the two, together with the anaphor, form a negative training instance. A classifier is trained on all training instances, using a standard supervised learning algorithm. During testing, all preceding markables of a candidate anaphor are considered as potential antecedents, and are tested in a back-to-front manner. The process stops if either an antecedent is found or the beginning of the text is reached. This framework has been widely used in the community of coreference resolution.

Recent work boosted the performance of coreference resolution by exploiting fine-tuned feature sets under the above framework, or adopting alternative resolution methods during testing (Ng and Cardie, 2002b; Yang et al., 2003; Denis and Baldridge, 2007; Versley et al., 2008).

Ng (2005) proposed a ranking model to maximize F-measure during testing. In the approach, n different coreference outputs for each test text are generated, by varying four components in a coreference resolution system, i.e., the learning algorithm, the instance creation method, the feature set, and the clustering algorithm. An SVM-based ranker then picks the output that is likely to have the highest F-measure. However, this approach is time-consuming during testing, as F-measure maximization is performed during testing. This limits its usage on a very large corpus.

In the community of machine learning, researchers have proposed approaches for learning a model to optimize a chosen evaluation metric other than classification accuracy on all training instances. Joachims (2005) suggested the use of support vector machines to optimize nonlinear evaluation metrics. However, the approach does not differentiate between the errors in the same category in the contingency table. Furthermore, it does not take into account inter-instance relation (e.g., transitivity), which the evaluation metric for coreference resolution cares about.

Daume III (2006) proposed a structured learning framework for coreference resolution to approximately optimize the ACE metric. Our proposed approach differs in two aspects. First, we directly optimize the evaluation metric itself, and not by approximation. Second, unlike the incremental local loss in Daume III (2006), we evaluate the metric score globally.

In contrast to Ng (2005), Ng and Cardie (2002a) proposed a rule-induction system with rule pruning. However, their approach is specific to rule induction, and is not applicable to other supervised learning classifiers. Ng (2004) varied different components of coreference resolution, choosing the combination of components that results in a classifier with the highest F-measure on a held-out development set during training. In contrast, our proposed approach employs instance weighting and beam search to maximize the F-measure of the evaluation metric during training. Our approach is general and applicable to any supervised learning classifiers.

Recently, Wick and McCallum (2009) proposed a partition-wise model for coreference resolution to maximize a chosen evaluation metric using the Metropolis-Hastings algorithm (Metropolis et al., 1953; Hastings, 1970). However, they found that training on classification accuracy, in most cases, outperformed training on the coreference evaluation metrics. Furthermore, similar to Ng (2005), their approach requires the generation of multiple coreference assignments during testing.

Vemulapalli *et al.* (2009) proposed a document-level boosting technique for coreference resolution by re-weighting the documents that have the lowest F-measures. By combining multiple classifiers generated in multiple iterations, they

achieved a CEAF score slightly better than the baseline. Different from them, our approach works at the instance level, and we output a single classifier.

3 Coreference Evaluation Metrics

In this section, we review two commonly used evaluation metrics for coreference resolution.

First, we introduce the terminology. The gold standard annotation and the output by a coreference resolution system are called key and response, respectively. In both the key and the response, a coreference chain is formed by a set of coreferential mentions. A *mention* (or markable) is a noun phrase which satisfies the markable definition in an individual corpus. A *link* refers to a pair of coreferential mentions. If a mention has no links to other mentions, it is called a *singleton*.

3.1 The MUC Evaluation Metric

Vilain *et al.* (1995) introduced the link-based MUC evaluation metric for the MUC-6 and MUC-7 coreference tasks. Let S_i be an equivalence class generated by the key (i.e., S_i is a coreference chain), and $p(S_i)$ be a partition of S_i relative to the response. Recall is the number of correctly identified links over the number of links in the key.

$$Recall = \frac{\sum(|S_i| - |p(S_i)|)}{\sum(|S_i| - 1)}$$

Precision, on the other hand, is defined in the opposite way by switching the role of key and response. F-measure is a trade-off between recall and precision.

$$F = \frac{2 \cdot Recall \cdot Precision}{Recall + Precision}$$

3.2 The B-CUBED Evaluation Metric

Bagga and Baldwin (1998) introduced the mention-based B-CUBED metric. The B-CUBED metric measures the accuracy of coreference resolution based on individual mentions. Hence, it also gives credit to the identification of singletons, which the MUC metric does not. Recall is computed as

$$Recall = \frac{1}{N} \sum_{d \in D} \sum_{m \in d} \frac{|O_m|}{|S_m|}$$

where D, d, and m are the set of documents, a document, and a mention, respectively. S_m is the equivalence class generated by the key that contains m, while O_m is the overlap of S_m and the equivalence class generated by the response that contains m. N is the total number of mentions in D. The precision, again, is computed by switching the role of key and response. F-measure is computed in the same way as the MUC metric.

4 Maximum Metric Score Training

Before explaining the algorithm, we describe our coreference clustering method used during testing. It is the same as most prior work in the literature, including Soon *et al.* (2001) and Ng and Cardie (2002b). The individual classification decisions made by the coreference classifier do not guarantee that transitivity of coreferential NPs is obeyed. So it can happen that the pair A and B, and the pair B and C are both classified as coreferential, but the pair A and C is not classified as coreferential by the classifier. After all coreferential markable pairs are found (no matter by closest-first, best-first, or resolving-all strategies as in different prior work), all coreferential pairs are clustered together to form the coreference output. By doing so, transitivity is kept: a markable is in a coreference chain if and only if it is classified to be coreferential to at least one other markable in the chain.

4.1 Instance Weighting

Suppose there are m_k and m_r coreferential links in the key and the response, respectively, and a coreference resolution system successfully predicts n correct links. The recall and the precision are then $\frac{n}{m_k}$ and $\frac{n}{m_r}$, respectively. The learnt classifier predicts false positive and false negative instances during testing. For a false positive instance, if we could successfully predict it as negative, the recall is unchanged, but the precision will be $\frac{n}{m_r - 1}$, which is higher than the original precision $\frac{n}{m_r}$. For a false negative instance, it is more subtle. If the two markables in the instance are determined to be in the same coreference chain by the clustering algorithm, it does not matter whether we predict this instance as positive or negative, i.e., this false negative does not

change the F-measure of the evaluation metric at all. If the two markables are not in the same coreference chain under the clustering, in case that we can predict it as positive, the recall will be $\frac{n+1}{m_k}$, which is higher than the original recall $\frac{n}{m_k}$, and the precision will be $\frac{n+1}{m_r+1}$, which is higher than the original precision $\frac{n}{m_r}$, as $n < m_r$. In both cases, the F-measure improves. If we can instruct the learning algorithm to pay more attention to these false positive and false negative instances and to predict them correctly by assigning them more weight, we should be able to improve the F-measure.

In the literature, besides the training instance extraction methods proposed by Soon *et al.* (2001) and Ng and Cardie (2002b) as discussed in Section 2, McCarthy and Lehnert (1995) used all possible pairs of training instances. We also use all pairs of training instances in our approach to keep as much information as possible. Initially all the pairs are equally weighted. We then iteratively assign more weights to the hard-to-classify pairs. The iterative process is conducted by a beam search algorithm.

4.2 Beam Search

Our proposed MMST algorithm searches for a set of weights to assign to training instances such that the classifier trained on the weighted training instances gives the maximum coreference metric score when evaluated on the training instances. Beam search is used to limit the search. Each search state corresponds to a set of weighted training instances, a classifier trained on the weighted training instances minimizing misclassifications, and the F-measure of the classifier when evaluated on the weighted training instances using the chosen coreference evaluation metric. The root of the search tree is the initial search state where all the training instances have identical weights of one. Each search state s can expand into two different children search states s_l and s_r. s_l (s_r) corresponds to assigning higher weights to the false positive (negative) training instances in s. The search space thus forms a binary search tree.

Figure 1 shows an example of a binary search tree. Initially, the tree has only one node: the root (node 1 in the figure). In each iteration, the algorithm expands all the leaf nodes in the beam. For example, in the first iteration, node 1 is expanded to generate node 2 and 3, which corresponds to adding weights to false positive and false negative training instances, respectively. An expanded node always has two children in the binary search tree. All the nodes are then sorted in descending order of F-measure. Only the top M nodes are kept, and the remaining nodes are discarded. The discarded nodes can either be leaf nodes or non-leaf nodes. For example, if node 5 is discarded because of low F-measure, it will not be expanded to generate children in the binary search tree. The iterative algorithm stops when all the nodes in the beam are non-leaf nodes, i.e., all the nodes in the beam have been expanded.

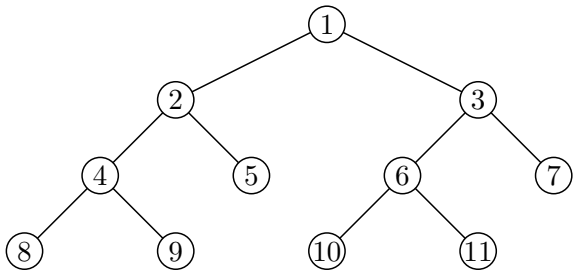


Figure 1: An example of a binary search tree

Figure 2 gives the formal description of the proposed maximum metric score training algorithm. In the algorithm, assume that we have N texts T_1, T_2, ..., T_N in the training data set. m_{ki} and m_{kj} are the ith and jth markable in the text T_k, respectively. Hence, for all $i < j$, $(m_{ki}, m_{kj}, w_{kij})$ is a training instance for the markable pair (m_{ki}, m_{kj}), in which w_{kij} is the weight of the instance. Let L_{kij} and L'_{kij} be the true and predicted label of the pair (m_{ki}, m_{kj}), respectively. Let W, C, F, and E be the set of weights $\{w_{kij}|1 \leq k \leq N, i < j\}$, the classifier, the F-measure, and a boolean indicator of whether the search state has been expanded, respectively. Finally, M is the beam size, and δ controls how much we update the weights in each iteration.

Since we train the model on all possible pairs, during testing we also test if a potential anaphor is coreferential to each preceding antecedent.

INPUT: $T_1, T_2, \ldots, T_N$
OUTPUT: classifier C
$w_{kij} \leftarrow 1$, for all $1 \leq k \leq N$ and $i < j$
$C \leftarrow$ train($\{(m_{ki}, m_{kj}, w_{kij}) | 1 \leq k \leq N, i < j\}$)
$F \leftarrow$ resolve and evaluate $T_1, \ldots, T_N$ with C
$E \leftarrow false$
BEAM$\leftarrow \{(W, C, F, E)\}$
repeat
  BEAM$' \leftarrow \{\}$
  for all (W, C, F, E) in BEAM **do**
    BEAM$' \leftarrow$ BEAM$' \bigcup \{(W, C, F, true)\}$
    if E=false **then**
      predict all L'_{kij} with C $(1 \leq k \leq N, i < j)$
      cluster into coreference chains based on L'_{kij}
      $W' \leftarrow W$
      for all $1 \leq k \leq N, i < j$ **do**
        if $L_{kij} = false$ and $L'_{kij} = true$ **then**
          $w'_{kij} \leftarrow w'_{kij} + \delta$
        end if
      end for
      $C' \leftarrow$ train($\{(m_{ki}, m_{kj}, w'_{kij}) | 1 \leq k \leq N, i < j\}$)
      $F' \leftarrow$ resolve and evaluate $T_1, \ldots, T_N$ with C'
      BEAM$' \leftarrow$ BEAM$' \bigcup \{(W', C', F', false)\}$
      $W'' \leftarrow W$
      for all $1 \leq k \leq N, i < j$ **do**
        if $L_{kij} = true$ and $L'_{kij} = false$ and
          $Chain(m_{ki}) \neq Chain(m_{kj})$ **then**
          $w''_{kij} \leftarrow w''_{kij} + \delta$
        end if
      end for
      $C'' \leftarrow$ train($\{(m_{ki}, m_{kj}, w''_{kij}) | 1 \leq k \leq N, i < j\}$)
      $F'' \leftarrow$ resolve and evaluate $T_1, \ldots, T_N$ with C''
      BEAM$' \leftarrow$ BEAM$' \bigcup \{(W'', C'', F'', false)\}$
    end if
  end for
  BEAM$\leftarrow$ BEAM$'$
  sort BEAM in descending order of F, keep top M elements
until for all E of all elements in BEAM, $E = true$
return C, from the top element (W, C, F, E) of BEAM

Figure 2: The maximum metric score training (MMST) algorithm

5 Experiments

5.1 Experimental Setup

In the experiments, we used all the five commonly used evaluation corpora for coreference resolution, namely the two MUC corpora (MUC6 and MUC7) and the three ACE corpora (BNEWS, NPAPER, and NWIRE). The MUC6 and the MUC7 corpora were defined in the DARPA Message Understanding Conference (MUC-6, 1995; MUC-7, 1998). The dry-run texts were used as the training data sets. In both corpora, each training data set contains 30 texts. The test data sets for MUC6 and MUC7 consist of the 30 and 20 formal evaluation texts, respectively. The ACE corpora were defined in NIST Automatic Content Extraction phase 2 (ACE-2) (NIST, 2002). The three data sets are from different news sources: broadcast news (BNEWS), newspaper (NPAPER), and newswire (NWIRE). Each of the three data sets contains two portions: training and development test. They were used as our training set and test set, respectively. The BNEWS, NPAPER, and NWIRE data sets contain 216, 76, and 130 training texts, and 51, 17, and 29 test texts, respectively.

Unlike some previous work on coreference resolution that assumes that the gold standard markables are known, we work directly on raw text input. Versley *et al.* (2008) presented the BART package[1], an open source coreference resolution toolkit, that accepts raw text input and reported state-of-the-art MUC F-measures on the three ACE corpora. BART uses an extended feature set and tree kernel support vector machines (SVM) under the Soon *et al.* (2001) training and testing framework. We used the BART package in our experiments, and implemented the proposed MMST algorithm on top of it. In our experiments reported in this paper, the features we used are *identical* to the features output by the preprocessing code of BART reported in Versley *et al.* (2008), except that we did not use their tree-valued and string-valued features (see the next subsection for details).

Since we use automatically extracted markables, it is possible that some extracted markables and the gold standard markables are unmatched, or *twinless* as defined in Stoyanov *et al.* (2009). How to use the B-CUBED metric for evaluating twinless markables has been explored recently. In this paper, we adopt the B^3all variation proposed by Stoyanov *et al.* (2009), which retains all twinless markables. We also experimented with their B^30 variation, which gave similar results. Note that no matter which variant of the B-CUBED metric is used, it is a fair comparison as long as the baseline and our proposed MMST algorithm are compared against each other using the same B-CUBED variant.

5.2 The Baseline Systems

We include state-of-the-art coreference resolution systems in the literature for comparison. Since we use the BART package in our experiments,

[1] `http://www.sfs.uni-tuebingen.de/~versley/BART/`

we include the results of the original BART system (with its extended feature set and SVM-light-TK (Moschitti, 2006), as reported in Versley *et al.* (2008)) as the first system for comparison. Versley *et al.* (2008) reported only the results on the three ACE data sets with the MUC evaluation metric. Since we used all the five data sets in our experiments, for fair comparison, we also include the MUC results reported in Ng (2004). To the best of our knowledge, Ng (2004) was the only prior work which reported MUC metric scores on all the five data sets. The MUC metric scores of Versley *et al.* (2008) and Ng (2004) are listed in the row "Versley *et al.* 08" and "Ng 04", respectively, in Table 1. For the B-CUBED metric, we include Ng (2005) for comparison, although it is unclear how Ng (2005) interpreted the B-CUBED metric. The scores are listed in the row "Ng 05" in Table 2.

Tree kernel SVM learning is time-consuming. To reduce the training time needed, instead of using SVM-light-TK, we used a much faster learning algorithm, J48, which is the WEKA implementation of the C4.5 decision tree learning algorithm. (Quinlan, 1993; Witten and Frank, 2005). As tree-valued features and string-valued features cannot be used with J48, in our experiments we excluded them from the extended feature set that BART used to produce state-of-the-art MUC F-measures on the three ACE corpora. All our results in this paper were obtained using this reduced feature set and J48 decision tree learning. However, given sufficient computational resources, our proposed approach is able to apply to any supervised machine learning algorithms.

Our baselines that follow the Soon *et al.* (2001) framework, using the reduced feature set and J48 decision tree learning, are shown in the row "*SNL*-Style Baseline" in Table 1 and 2. The results suggest that our baseline system is comparable to the state of the art. Although in Table 1, the performance of the *SNL*-style baseline is slightly lower than Versley *et al.* (2008) on the three ACE corpora, the computational time needed has been greatly reduced.

Our MMST algorithm trains and tests on all pairs of markables. To show the effectiveness of weight updating of MMST, we built another baseline which trains and tests on all pairs. The performance of this system is shown in the row "*All*-Style Baseline" in Table 1 and 2.

5.3 Results Using Maximum Metric Score Training

Next, we show the results of using the proposed maximum metric score training algorithm. From the description of the algorithm, it can be seen that there are two parameters in the algorithm. One parameter is M, the size of the beam. The other parameter is δ, which controls how much we increase the weight of a training instance in each iteration.

Since the best M and δ for the MUC evaluation metric were not known, we used held-out development sets to tune the parameters. Specifically, we trained classifiers with different combinations of M and δ on a development training set, and evaluated their performances on a development test set. In our experiments, the development training set contained 2/3 of the texts in the training set of each individual corpus, while the development test set contained the remaining 1/3 of the texts. After having picked the best M and δ values, we trained a classifier on the entire training set with the chosen parameters. The learnt classifier was then applied to the test set.

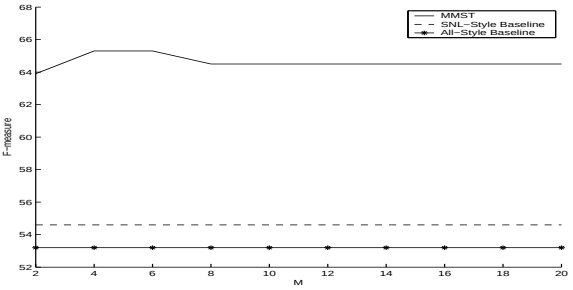


Figure 3: Tuning M on the held-out development set

To limit the search space, we tuned the two parameters sequentially. First, we fixed $\delta = 1$, which is equivalent to duplicating each training instance once in J48, and evaluated $M = 2, 4, 6, \ldots, 20$. After having chosen the best M that corresponded to the maximum F-measure, we fixed the value of M, and evaluated $\delta = 0.1, 0.2, 0.3, \ldots, 2.0$. Take MUC6 as an exam-

Model	MUC6			MUC7			BNEWS			NPAPER			NWIRE		
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
Versley *et al.* 08	–			–			60.7	65.4	63.0	64.1	67.7	65.8	60.4	65.2	62.7
Ng 04	75.8	61.4	**67.9**	64.2	60.2	62.1	63.1	67.8	65.4	73.5	63.3	**68.0**	53.1	60.6	56.6
SNL-Style Baseline	67.0	49.2	56.7	63.0	54.2	58.3	57.4	64.3	60.7	61.6	67.3	64.3	58.6	66.1	62.1
All-Style Baseline	56.9	69.2	62.5	51.5	73.4	60.6	53.0	76.7	62.7	56.3	75.4	64.4	53.0	74.5	61.9
MMST	73.3	59.9	65.9**††	66.8	59.8	**63.1****††	70.5	61.9	**65.9****†	69.9	64.0	66.8†	64.7	64.7	**64.7****†
	$M = 6,\ \delta = 1.0$			$M = 6,\ \delta = 0.7$			$M = 6,\ \delta = 1.8$			$M = 6,\ \delta = 0.9$			$M = 14,\ \delta = 0.7$		

Table 1: Results for the two MUC and three ACE corpora with MUC evaluation metric

Model	MUC6			MUC7			BNEWS			NPAPER			NWIRE		
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
Ng 05	–			–			57.0	77.1	65.6	62.8	71.2	66.7	59.3	75.4	66.4
SNL-Style Baseline	57.8	74.4	65.1	57.6	76.5	65.7	62.0	74.7	67.8	61.8	70.4	65.8	65.8	75.9	70.5
All-Style Baseline	51.6	86.3	64.6	49.1	90.1	63.6	61.6	83.7	**71.0**	63.9	74.0	68.6	64.8	80.1	**71.7**
MMST	62.7	81.5	**70.9****††	61.8	73.6	**67.2**††	61.6	83.7	**71.0****	63.1	76.2	**69.1****	64.3	81.0	**71.7**
	$M = 6,\ \delta = 1.0$			$M = 8,\ \delta = 0.8$			$M = 6,\ \delta = 0.9$			$M = 14,\ \delta = 0.5$			$M = 6,\ \delta = 0.1$		

Table 2: Results for the two MUC and three ACE corpora with B^3 evaluation metric

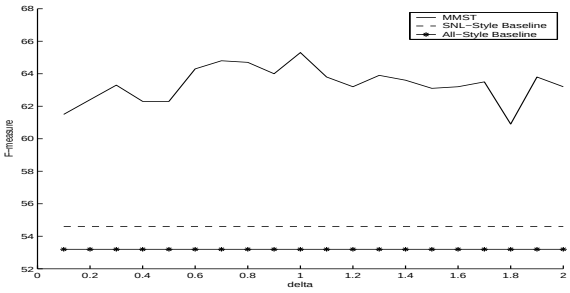


Figure 4: Tuning δ on the held-out development set

ple. The results of tuning M on MUC6 are shown in Figure 3. The maximum F-measure is obtained when $M = 4$ and $M = 6$. On all the different M values we have tried, MMST outperforms both the *SNL*-style baseline and the *All*-style baseline on the development test set. We then fixed $M = 6$, and evaluated different δ values. The results are shown in Figure 4. The best F-measure was obtained when $\delta = 1.0$. Again, on all the different δ values we have tried, MMST outperforms both baselines on the development test set.

The rows "MMST" in Table 1 and 2 show the performance of MMST on the test sets, with the tuned parameters indicated. In our experiments, the statistical significance test was conducted as in Chinchor (1995). * and ** stand for $p < 0.05$ and $p < 0.01$ over the *SNL*-style baseline, respectively. † and †† stand for $p < 0.05$ and $p < 0.01$ over the *All*-style baseline, respectively.

For the MUC metric, when compared to the *All*-style baseline, MMST gains 3.4, 2.5, 3.2, 2.4, and 2.8 improvement in F-measure on MUC6, MUC7, BNEWS, NPAPER, and NWIRE, respectively. The experimental results clearly show that MMST gains not only consistent, but also statistically significant improvement over both the *SNL*-style baseline and the *All*-style baseline in all combinations (five data sets and two baselines) on the MUC metric, except that it is not significant ($p = 0.06$) over the *SNL*-style baseline in NPAPER. As for the B-CUBED metric, MMST gains significant improvement in F-measure on MUC6 and MUC7 data sets, while its performance on the three ACE data sets are comparable to the *All*-style baseline.

5.4 Discussion

To see how MMST actually updates the weight, we use the MUC metric as an example. Under the experimental settings, it takes 6 – 9 iterations for MMST to stop on the five data sets. The number of explored states in the binary search tree, including the root, is 33, 39, 25, 29, and 75 on MUC6, MUC7, BNEWS, NPAPER, and NWIRE, respectively. It is instructive to find out the final weight of each instance. Take MUC6 as an example, the number of positive instances with weight 1, 2, 3, and 4 are 5,204, 1,568, 1,379, and 1,844, respectively, while the number of negative instances with weight 1 and 2 are 503,141 and 1,755, respec-

tively. Counting the weighted number of instances (e.g., an instance with weight 2 is equivalent to 2 instances), we have 19,853 positive and 506,651 negative training instances. This changes the ratio of the positive instances from 1.9% to 3.8%. As a by-product, MMST reduces data skewness, while using all possible NP pairs for training to keep as much information as possible.

The change of weights of the training instances is equivalent to the change of distribution of the training instances. This effectively changes the classification hypothesis to the one that tends to yield higher evaluation metric score. Take the following sentence in the MUC6 data set as an example:

> In a news release, ***the company*** said the new name more accurately reflects ***its*** focus on high-technology communications, including business and entertainment software, interactive media and wireless data and voice transmission.

In the above example, the pronoun *its* is coreferential to the antecedent NP *the company*. The baseline classifier gives a probability of 0.02 that the two NPs are coreferential. The pair is classified wrongly and none of the other pairs in the article can link the two NPs together through clustering. However, with MMST, this probability increases to 0.54, which leads to the correct classification. This is because the baseline classifier is not good at predicting in the case when the second markable is a pronoun. In the above example, *its* can have another candidate antecedent *the new name*. There are far more negative training instances than positive ones for this case. In fact, in the induced decision tree by the baseline, the leaf node corresponding to the pair *the company – its* has 7,782 training instances, out of which only 175 are positive. With MMST, however, these numbers decrease to 83 and 45, respectively. MMST also promotes the Anaphor_Is_Pronoun feature to a higher level in the decision tree. Although we use decision tree to illustrate the working of the algorithm, MMST is not limited to tree learning, and can make use of any learning algorithms that are able to take advantage of instance weighting.

It can also be seen that with the B-CUBED metric, MMST gains improvement on MUC6 and MUC7, but not on the three ACE corpora. However, the results of MMST on the three ACE corpora with the B-CUBED evaluation metric are at least comparable with the *All*-style baseline. This is because we always pick the classifier which corresponds to the maximum evaluation metric score on the training set and the classifier corresponding to the *All*-style baseline is one of the candidates. In addition, our MMST approach improves upon state-of-the-art results (Ng, 2004; Ng, 2005; Versley et al., 2008) on most of the five standard benchmark corpora (two MUC corpora and three ACE corpora), with both the link-based MUC metric and the mention-based B-CUBED metric.

Finally, our approach performs all the F-measure maximization during training, and is very fast during testing, since the output of the MMST algorithm is a standard classifier. For example, on the MUC6 data set with the MUC evaluation metric, it took 1.6 hours and 31 seconds for training and testing, respectively, on an Intel Xeon 2.33GHz machine.

6 Conclusion

In this paper, we present a novel maximum metric score training approach comprising the use of instance weighting and beam search to maximize the chosen coreference metric score on the training corpus during training. Experimental results show that the approach achieves significant improvement over the baseline systems. The proposed approach improves upon state-of-the-art results on most of the five standard benchmark corpora (two MUC corpora and three ACE corpora), with both the link-based MUC metric and the mention-based B-CUBED metric.

Acknowledgments

We thank Yannick Versley for providing us the BART package and the preprocessed data. This research was done for CSIDM Project No. CSIDM-200804 partially funded by a grant from the National Research Foundation (NRF) administered by the Media Development Authority (MDA) of Singapore.

References

Bagga, Amit and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *Proceedings of the LREC1998*, pages 563–566.

Chinchor, Nancy. 1995. Statistical significance of MUC-6 results. In *Proceedings of the MUC-6*, pages 39–43.

Daume III, Hal. 2006. *Practical Structured Learning for Natural Language Processing*. Ph.D. thesis, University of Southern California.

Denis, Pascal and Jason Baldridge. 2007. Joint determination of anaphoricity and coreference resolution using integer programming. In *Proceedings of the NAACL-HLT2007*, pages 236–243.

Hastings, W. K. 1970. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109.

Joachims, Thorsten. 2005. A support vector method for multivariate performance measures. In *Proceedings of the ICML2005*, pages 377–384.

McCarthy, Joseph F. and Wendy G. Lehnert. 1995. Using decision trees for coreference resolution. In *Proceedings of the IJCAI1995*, pages 1050–1055.

Metropolis, Nicholas, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. 1953. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1092.

Moschitti, Alessandro. 2006. Making tree kernels practical for natural language learning. In *Proceedings of the EACL2006*, pages 113–120.

MUC-6. 1995. Coreference task definition (v2.3, 8 Sep 95). In *Proceedings of the MUC-6*, pages 335–344.

MUC-7. 1998. Coreference task definition (v3.0, 13 Jul 97). In *Proceedings of the MUC-7*.

Ng, Vincent and Claire Cardie. 2002a. Combining sample selection and error-driven pruning for machine learning of coreference rules. In *Proceedings of the EMNLP2002*, pages 55–62.

Ng, Vincent and Claire Cardie. 2002b. Improving machine learning approaches to coreference resolution. In *Proceedings of the ACL2002*, pages 104–111.

Ng, Vincent. 2004. *Improving Machine Learning Approaches to Noun Phrase Coreference Resolution*. Ph.D. thesis, Cornell University.

Ng, Vincent. 2005. Machine learning for coreference resolution: From local classification to global ranking. In *Proceedings of the ACL2005*, pages 157–164.

NIST. 2002. The ACE 2002 evaluation plan. `ftp://jaguar.ncsl.nist.gov/ace/doc/ACE-EvalPlan-2002-v06.pdf`.

Quinlan, J. Ross. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Soon, Wee Meng, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

Stoyanov, Veselin, Nathan Gilbert, Claire Cardie, and Ellen Riloff. 2009. Conundrums in noun phrase coreference resolution: Making sense of the state-of-the-art. In *Proceedings of the ACL-IJCNLP2009*, pages 656–664.

Vemulapalli, Smita, Xiaoqiang Luo, John F. Pitrelli, and Imed Zitouni. 2009. Classifier combination techniques applied to coreference resolution. In *Proceedings of the NAACL-HLT2009 Student Research Workshop and Doctoral Consortium*, pages 1–6.

Versley, Yannick, Simone Paolo Ponzetto, Massimo Poesio, Vladimir Eidelman, Alan Jern, Jason Smith, Xiaofeng Yang, and Alessandro Moschitti. 2008. BART: A modular toolkit for coreference resolution. In *Proceedings of the ACL2008:HLT Demo Session*, pages 9–12.

Vilain, Marc, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the MUC-6*, pages 45–52.

Wick, Michael and Andrew McCallum. 2009. Advances in learning and inference for partition-wise models of coreference resolution. Technical Report UM-CS-2009-028, University of Massachusets, Amherst, USA.

Witten, Ian H. and Eibe Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann Publishers, second edition.

Yang, Xiaofeng, Guodong Zhou, Jian Su, and Chew Lim Tan. 2003. Coreference resolution using competition learning approach. In *Proceedings of the ACL2003*, pages 176–183.

Paraphrasing with Search Engine Query Logs

Shiqi Zhao[†‡], **Haifeng Wang**[†], **and Ting Liu**[‡]
[†]Baidu Inc.
[‡]HIT Center for Information Retrieval, Harbin Institute of Technology
{zhaoshiqi, wanghaifeng}@baidu.com, tliu@ir.hit.edu.cn

Abstract

This paper proposes a method that extracts paraphrases from search engine query logs. The method first extracts paraphrase query-title pairs based on an assumption that a search query and its corresponding clicked document titles may mean the same thing. It then extracts paraphrase query-query and title-title pairs from the query-title paraphrases with a pivot approach. Paraphrases extracted in each step are validated with a binary classifier. We evaluate the method using a query log from Baidu[1], a Chinese search engine. Experimental results show that the proposed method is effective, which extracts more than 3.5 million pairs of paraphrases with a precision of over 70%. The results also show that the extracted paraphrases can be used to generate high-quality paraphrase patterns.

1 Introduction

The use of paraphrases is ubiquitous in human languages, which also presents a challenge for natural language processing (NLP). Previous studies have shown that paraphrasing can play important roles in plenty of areas, such as machine translation (MT) (Callison-Burch et al., 2006; Kauchak and Barzilay, 2006), question answering (QA) (Duboue and Chu-Carroll, 2006; Riezler et al., 2007), natural language generation (NLG) (Iordanskaja et al., 1991), and so on. As a result, the research on paraphrasing and its applications have attracted significant interest.

[1]www.baidu.com

This paper proposes a method that uses search engine query logs for extracting paraphrases, which is illustrated in Figure 1. Specifically, three kinds of paraphrases can be extracted with our method, which include (1) query-title (Q-T): a query and a document title that users clicked on; (2) query-query (Q-Q): two queries, for which users clicked on the same document title; (3) title-title (T-T): two titles that users clicked on for the same query. We train a classifier for each kind to filter incorrect pairs and refine the paraphrases.

Extracting paraphrases using query logs has many advantages. First, query logs keep growing, which have no scale limitation. Second, query logs reflect web users' real needs, hence the extracted paraphrases may be more useful than that from other kinds of corpora. Third, paraphrases extracted from query logs can be directed applied in search engines for query suggestion and document reranking. In addition, we find that both queries and titles contain a good many question sentences, which can be useful in developing QA systems.

We conduct experiments using a query log of a commercial Chinese search engine Baidu, from which we extracted about 2.7 million pairs of paraphrase Q-T, 0.4 million pairs of paraphrase Q-Q, and 0.4 million pairs of paraphrase T-T. The precision of the paraphrases is above 70%. In addition, we generate paraphrase patterns using the extracted paraphrases. The results show that 73,484 pairs of paraphrase patterns have been generated, with a precision of over 78%.

In the rest of the paper, we first review related work in Section 2. Section 3 describes our method in detail. Section 4 presents the evaluation and re-

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1317–1325,
Beijing, August 2010

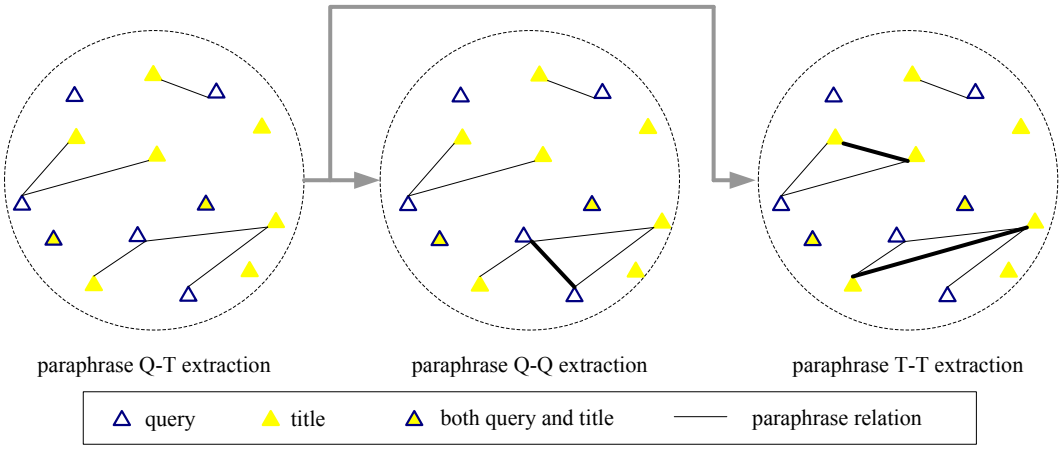


Figure 1: Illustration of the proposed method.

sults. Section 5 concludes the paper and discusses future directions.

2 Related Work

In this section, we briefly review previous studies on paraphrase extraction and query log mining in information retrieval (IR).

2.1 Paraphrase Extraction

A variety of data resources have been exploited for paraphrase extraction. For example, some researchers extract paraphrases from multiple translations of the same foreign novel (Barzilay and McKeown, 2001; Ibrahim et al., 2003), while some others make use of comparable news articles that report on the same event within a small time interval (Shinyama et al., 2002; Barzilay and Lee, 2003; Dolan et al., 2004). Besides the monolingual corpora, bilingual parallel corpora have also been used for extracting paraphrases (Bannard and Callison-Burch, 2005; Callison-Burch, 2008; Zhao et al., 2008). Their basic assumption is that phrases that align with the same foreign phrase may have the same meaning.

The above methods have achieved promising results. However, their performances are usually constrained due to the scale and domain limitation. As an alternative, researchers have tried to acquire paraphrases from large-scale web corpora (Lin and Pantel, 2001; Paşca and Dienes, 2005; Bhagat and Ravichandran, 2008) or directly based on web mining (Ravichandran and Hovy, 2002). These methods are guided by an extended version of distributional hypothesis, namely, if two phrases often occur in similar contexts, their meanings tend to be similar. The disadvantage of these methods is that the underlying assumption does not always hold. Phrases with opposite meanings can also occur in similar contexts, such as "X solves Y" and "X worsens Y" (Lin and Pantel, 2001). In addition, the extracted paraphrases are generally short fragments with two slots (variables) at both ends.

2.2 Query Log Mining in IR

Query logs are widely used in the IR community, especially for mining similar queries. For example, Wen et al. (2002) clustered queries based on user click information. Their basic idea is that if some queries result in similar user clicks, the meanings of these queries should be similar. Such methods have also been investigated in (Gao et al., 2007) for cross-lingual query suggestion and (Zhao et al., 2007) for synonymous questions identification. This paper is partly inspired by their studies. However, we do not simply use click information as clues for mining similar queries. Instead, we mine paraphrases across queries and clicked document titles.

In addition, query logs can be used for query expansion. For instance, Cui et al. (2002) extract probabilistic correlations between query terms and document terms by analyzing query logs, which are then used to select high-quality

H1:	If a query q hits a title t, then q and t are likely to be paraphrases.
H2:	If queries q_1 and q_2 hit the same title t, q_1 and q_2 are likely to be paraphrases.
H3:	If a query q hits titles t_1 and t_2, then t_1 and t_2 are likely to be paraphrases.

Table 1: Hypotheses for extracting paraphrases.

expansion terms for new queries. Note that the expansion terms are merely related terms of the queries, not necessarily paraphrases.

There are other studies that use query logs for constructing ontologies (Sekine and Suzuki, 2007), learning named entities (Paşca, 2007), building user profiles (Richardson, 2008), correcting spelling errors (Ahmad and Kondrak, 2005), and so forth.

3 The Proposed Method

3.1 Basic Idea

Nowadays, more and more users tend to search long queries with search engines. Many users even directly search questions to get exact answers. By analyzing our query log that records rich information including user queries, clicked urls, titles, etc., we find that most titles of clicked documents are highly related with search queries. Especially, paraphrases can be easily found from long queries and the corresponding clicked titles. This motivates us to extract paraphrases from query-title pairs. Here we introduce a concept *hit* that will be frequently used: given a query q, a web document d, and d's title t, if there exist some users that click on d when searching q, then we say q *hits* t.

The hypothesis for extracting paraphrase Q-T is shown in Table 1 (H1). In addition, we find that when several queries hit the same title, the queries are likely to be paraphrases of each other. The other way round, when a query hits several titles, paraphrases can also be found among the titles. We therefore further extract paraphrase Q-Q and T-T from the paraphrase Q-T. The underlying hypotheses can be found in Table 1 (H2 and H3). Note that, based on H2 and H3, paraphrase Q-Q and T-T can be directly extracted from raw Q-T pairs. However, in consideration of precision, we extract them from paraphrase Q-T. We call our paraphrase Q-Q and T-T extraction approach as a pivot approach, since we use titles as *pivots* (queries as *targets*) when extracting paraphrase Q-Q and use queries as *pivots* (titles as *targets*) when extracting paraphrase T-T.

INPUT: $\mathcal{Q}$: query space, $\mathcal{T}$: title space
OUTPUT: P_{qt}: the set of paraphrase Q-T,
P_{qq}: the set of paraphrase Q-Q,
P_{tt}: the set of paraphrase T-T,
$ParaSet$: the set of paraphrases

1. **FOR** any $q \in \mathcal{Q}$ and $t \in \mathcal{T}$
2. **IF** q hits t
3. **IF** $IsParaphrase(q, t)$
4. Add $\langle q, t \rangle$ to P_{qt}
5. **END IF**
6. **END IF**
7. **END FOR**

8. **FOR** any $q_1, q_2 \in \mathcal{Q}$ and $t \in \mathcal{T}$
9. **IF** $\langle q_1, t \rangle \in P_{qt}$ and $\langle q_2, t \rangle \in P_{qt}$
10. **IF** $IsParaphrase(q_1, q_2)$
11. Add $\langle q_1, q_2 \rangle$ to P_{qq}
12. **END IF**
13. **END IF**
14. **END FOR**

15. **FOR** any $t_1, t_2 \in \mathcal{T}$ and $q \in \mathcal{Q}$
16. **IF** $\langle q, t_1 \rangle \in P_{qt}$ and $\langle q, t_2 \rangle \in P_{qt}$
17. **IF** $IsParaphrase(t_1, t_2)$
18. Add $\langle t_1, t_2 \rangle$ to P_{tt}
19. **END IF**
20. **END IF**
21. **END FOR**

22. **RETURN** $ParaSet = P_{qt} \cup P_{qq} \cup P_{tt}$

Table 2: Algorithm for extracting paraphrases.

3.2 Algorithm

Our paraphrase extraction algorithm is shown in Table 2. In particular, lines 1∼7 extract para-

phrase Q-T from the query log. Lines 8∼14 and 15∼21 extract paraphrase Q-Q and T-T, respectively. Line 22 combines the paraphrase Q-T, Q-Q, and T-T together. To filter noise, the extracted Q-T, Q-Q, and T-T pairs are all validated using a function $IsParaphrase(s_1, s_2)$. In this work, we recast paraphrase validation as a binary classification problem. Any pair of $\langle s_1, s_2 \rangle$ is classified as 1 (paraphrase) or 0 (non-paraphrase) with a support vector machine (SVM) classifier. The features used for classification will be detailed in Section 3.3.

In practice, we exploit a query log that contains 287 million Q-T pairs, which are then filtered using the following constraints: (1) exclude Q-T pairs that are too short, i.e., either query q or tittle t contains less than three terms; (2) exclude Q-T pairs where q subsumes t or vice versa, e.g., "牛肉 (beef)" and "牛肉 的 做法 (cooking method of beef)"; (3) exclude Q-T pairs in which the similarity between q and t is below a predefined threshold T[2]; (4) exclude Q-T pairs whose t contains frequent internet terms, such as "主页 (home page)", "网站 (web site)", "在线 (online)", since such titles are mostly organization home pages, online videos, downloadable resources, etc., which are useless for our purpose of paraphrase extraction.

3.3 Features for Paraphrase Validation

Given a pair of candidate paraphrases $\langle s_1, s_2 \rangle$, in which s_1 and s_2 can be either a query or a title, we exploit the following features in the classification-based paraphrase validation.

• **Frequency Feature** F_F. F_F is defined based on each $\langle s_1, s_2 \rangle$'s frequency. We expect that more frequent $\langle s_1, s_2 \rangle$ should be more reliable.

$$F_F(s_1, s_2) = \begin{cases} \frac{c(s_1,s_2)}{C} & if \;\; c(s_1, s_2) < C \\ 1 & if \;\; c(s_1, s_2) \geq C \end{cases} \quad (1)$$

where $c(s_1, s_2)$ denotes the number of times that the $\langle s_1, s_2 \rangle$ pair occurs in the corpus. C is a normalizing factor ($C = 10$ in our experiments).

[2]The similarity is computed based on word overlap rate, which will be described in detail in section 3.3. We set $T = 0.6$ in the experiments.

• **Length Rate Feature** F_{LR}:

$$F_{LR}(s_1, s_2) = \frac{min\{c_w(s_1), c_w(s_2)\}}{max\{c_w(s_1), c_w(s_2)\}} \quad (2)$$

where $c_w(s)$ denotes the number of words in s.

• **Word Overlap Rate Feature** F_{WOR}:

$$F_{WOR}(s_1, s_2) = \frac{c_w(s_1 \cap s_2)}{max\{c_w(s_1), c_w(s_2)\}} \quad (3)$$

where "$s_1 \cap s_2$" is the intersection of s_1 and s_2.

• **Character Overlap Rate Feature** F_{COR}. Chinese words are composed of characters. It is quite often that words with similar characters share similar meanings, such as "爽快 (comfortable)" and "痛快 (comfortable)", "出售 (sell)" and "销售 (sell)". Here we use F_{COR} to measure the similarity between s_1 and s_2 at the character level. Detailedly, we segment s_1 and s_2 into sets of characters and compute the overlap rate based on Equation (3)[3].

• **Cosine Similarity Feature** F_{CS}. In F_{CS}, both s_1 and s_2 are represented as vectors and their cosine similarity is computed as:

$$F_{CS}(s_1, s_2) = \frac{vec_w(s_1) \cdot vec_w(s_2)}{\|vec_w(s_1)\| \times \|vec_w(s_2)\|} \quad (4)$$

where $vec_w(s)$ is the vector of words in s, "$\cdot$" denotes the dot product of two vectors, $\|vec_w(s)\|$ is the norm of a vector. Here, the weight of each word w in a vector is computed using a heuristic similar to tf-idf:

$$W(w) = tf(w) \times log(\frac{N}{c(w)} + 0.1) \quad (5)$$

where $tf(w)$ is the frequency of w in the given s, $c(w)$ is the number of times that w occurs in the corpus, $N = \max_w c(w)$.

• **Edit Distance Feature** F_{ED}. Let $ED(s_1, s_2)$ be the edit distance at the word level between s_1 and s_2, we compute F_{ED} as follows:

$$F_{ED}(s_1, s_2) = 1 - \frac{ED(s_1, s_2)}{max\{c_w(s_1), c_w(s_2)\}} \quad (6)$$

[3]In F_{COR}, $c_w(s)$ of Equation (3) denotes the number of characters in s.

• **Named Entity (NE) Similarity Feature** F_{NE}. NE information is critical in paraphrase identification (Shinyama et al., 2002). We therefore compute the NE similarity between s_1 and s_2 and take it as a feature. We employ a Chinese NE recognition tool that can recognize *person names*, *locations*, *organizations*, and *numerals*. The NE similarity is computed as:

$$F_{NE}(s_1, s_2) = \frac{c_{ne}(s_1 \cap s_2) + 1}{max\{c_{ne}(s_1), c_{ne}(s_2)\} + 1} \quad (7)$$

where $c_{ne}(s)$ denotes the number of NEs in s. Equation (7) guarantees $F_{NE} = 1$ if there are no NEs in either s_1 or s_2.

• **Pivot Fertility Feature** F_{PF}**:** F_{PF} is a feature specially designed for paraphrase Q-Q and T-T extraction, which are based on the pivot approach[4]. Specifically, we define *fertility* of a pivot as the number of targets it corresponds to. Our observation indicates that the larger the fertility of a pivot is, the more noisy the targets are. Hence we define F_{PF} as:

$$F_{PF}(s_1, s_2) = \max_p \frac{1}{f(p)} \quad (8)$$

where $s_1 = q_1$, $s_2 = q_2$, $p = t$ when classifying Q-Q, while $s_1 = t_1$, $s_2 = t_2$, $p = q$ when classifying T-T. $f(p)$ denotes the fertility of the pivot p. The value is maximized over p if s_1 and s_2 can be extracted with multiple pivots.

3.4 Generating Paraphrase Patterns

A key feature of our method is that the extracted paraphrases are particularly suitable for generating paraphrase patterns, especially for the hot domains that are frequently searched. For example, there are quite a few paraphrases concerning the therapy of various diseases, from which we can easily induce patterns expressing the meaning of "How to treat [X] disease", such as "[X] 病 如何 治疗", "怎么 治疗 [X] 病", and "[X] 病 的 治疗 方法". Therefore, in this work, we try to generate paraphrase patterns using the extracted paraphrases.

In our preliminary experiments, we only induce paraphrase patterns from paraphrases that contain no more than 6 words. In addition, only one slot is allowed in each pair of paraphrase patterns. Let s_1 and s_2 be a pair of paraphrases extracted above. If there exist words $w \in s_1$ and $v \in s_2$ that satisfy (1) $w = v$, (2) w and v are not stop words, then we can induce a pair of paraphrase patterns by replacing w in s_1 and v in s_2 with a slot "[X]". It is obvious that several pairs of paraphrase patterns may be induced from one pair of paraphrases.

[4] F_{PF} is not used in paraphrase Q-T validation.

	SAME	RELA	DIFF
percent (%)	55.92	44.08	-

Table 3: Human labeling of candidate Q-T.

4 Experiments

We experiment with a query log that contains a total of 284,316,659 queries. Statistics reveal that 170,315,807 queries (59.90%) lead to at least one user click, each having 1.69 clicks on average. We extract 287,129,850 raw Q-T pairs using the query log, from which 4,448,347 pairs of candidate Q-T are left after filtering as described in Section 3.2. Almost all queries and titles are written in Chinese, though some of them contain English or Japanese words. The preprocessing of candidate Q-T includes Chinese word segmentation (WSeg) and NE recognition (NER). Our WSeg tool is implemented based on forward maximum matching, while the NER tool is based on a NE dictionary mined from the web.

4.1 Evaluation of Candidate Q-T

We first evaluate candidate Q-T without validation. To this end, we randomly sampled 5000 pairs of candidate Q-T and labeled them manually. Each pair is labeled into one of the 3 classes: SAME - q and t have the same meaning; RELA - q and t have related meanings; DIFF - q and t have clearly different meanings. The labeling results are listed in Table 3. We can see that no candidate Q-T is in the DIFF class. This is not surprising, since users are unlikely to click on web pages unrelated to their queries.

To gain a better insight into the data, we analyzed the subtle types of candidate Q-T in both SAME and RELA classes. In detail, we sampled

1000 pairs of candidate Q-T from the 5000 pairs labeled above, in which 563 are in the SAME class, while the other 437 are in the RELA class. Our analysis suggests that candidate Q-T in the SAME class can be divided into 4 subtle types:

- Trivial change (12.61%): changes of punctuation or stop words, such as "考研 失败 怎么 办" and "考研 失败 怎么 办 ？ ".
- Word or phrase replacement (68.38%): replacements of synonymous words or phrases, such as "咖啡 斑 的 治疗 多少 钱 (how mach is ...)" and "咖啡 斑 的 治疗 费用 是 多少 (what is the price of ...)".
- Structure change (7.10%): changes of both words and word orders, such as "减肥 中 水果 可以 吃 什么 (what fruit can I eat on a diet)" and "吃 什么 水果 可以 瘦身 (what fruit can help loss weight)".
- Others (11.90%): candidate Q-T that cannot be classified into the 3 types above.

The above analysis reveals that more than two thirds of candidate Q-T in the SAME class are in the "word or phrase replacement" type, while the ones with structure changes are slightly more than 7%. We believe this is mainly because queries and titles are relatively short and their structures are simple. Thus structure rewriting can hardly be conducted. This distribution is in line with that reported in (Zhao et al., 2008).

As for the RELA class, we find that 42.33% of such candidate Q-T share a problem of named entity mismatch, such as "美国 (US) 大型 水利 工程" and "中国 (China) 急需 大型 水利 工程". This indicates that the NE similarity feature is necessary in paraphrase validation.

4.2 Evaluation of Paraphrase Q-T

The candidate Q-T extracted above are classified with a SVM classifier[5] under its default setting. To evaluate the classifier, we run 5-fold cross validation with the 5000 human annotated data, in which we use 4000 for training and the rest 1000 for testing in each run. The evaluation criteria are precision (P), recall (R), and f-measure (F), which are defined as follows:

$$P = \frac{\|S_a \cap S_m\|}{\|S_a\|} \tag{9}$$

$$R = \frac{\|S_a \cap S_m\|}{\|S_m\|} \tag{10}$$

$$F = \frac{2 \times P \times R}{P + R} \tag{11}$$

where S_a is the set of paraphrases automatically recognized with the classifier, S_m is the set of paraphrases manually annotated. Precision, recall, and f-measure are averaged over 5 runs in the 5-fold cross validation.

Figure 2 (a) shows the classification results (dark bars). For comparison, we also show the precision, recall[6], and f-measure of the candidate Q-T (light bars). As can be seen, the precision is improved from 0.5592 to 0.7444 after classification. F-measure is also evidently enhanced. This result indicates that the classification-based paraphrase validation is effective. We then use all of the 5000 annotated data to train a classifier and classify all the candidate Q-T. Results show that 2,762,291 out of 4,448,347 pairs of candidate Q-T are classified as paraphrases.

4.3 Evaluation of Paraphrase Q-Q and T-T

From the paraphrase Q-T, we further extracted 934,758 pairs of candidate Q-Q and 438,954 pairs of candidate T-T (without validation). We randomly sampled 5000 from each for human annotation. The results show that the precisions of candidate Q-Q and T-T are 0.4672 and 0.6860, respectively. As can be seen, the precision of candidate Q-Q is much lower than that of candidate T-T. Our analysis reveals that it is mainly because candidate Q-Q are more noisy, since user queries contain quite a lot of spelling mistakes and informal expressions.

The candidate Q-Q and T-T are also refined based on classification. We first evaluate the classification performance using the 5000 human labeled data. The experimental setups for Q-Q and

[5]We use libsvm-2.82 toolkit, which can be downloaded from http://www.csie.ntu.edu.tw/ cjlin/libsvm/

[6]We assume all possible paraphrases are included in the candidates, thus its recall is 100%.

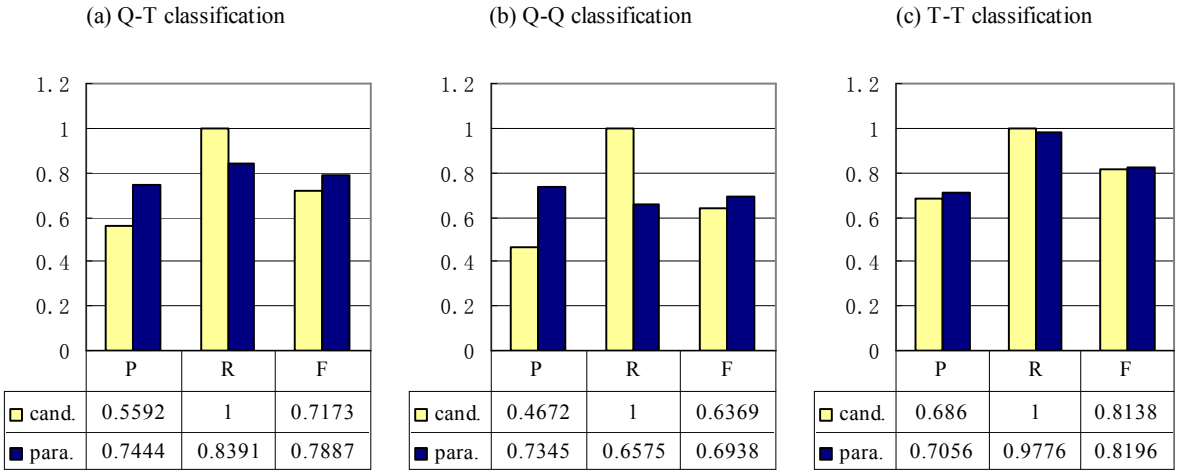


Figure 2: Classification precision (P), recall (R), and f-measure (F).

T-T classification are the same as that of Q-T classification, in which we run 5-fold cross validation with a SVM classifier using its default parameters. Figure 2 (b) and (c) give the classification results (dark bars) as well as the precision, recall, and f-measure of the candidates (light bars).

We can see that the precision of Q-Q is significantly enhanced from 0.4672 to 0.7345 after classification, which suggests that a substantial part of errors and noise are removed. The increase of f-measure demonstrates the effectiveness of classification despite the decrease of recall. Meanwhile, the quality of candidate T-T is not clearly improved after classification. The reason should be that the precision of candidate T-T is already pretty high. We then use all 5000 human labeled data to train a classifier for Q-Q and T-T respectively and classify all candidate Q-Q and T-T. Results show that 390,920 pairs of paraphrase Q-Q and 415,539 pairs of paraphrase T-T are extracted after classification.

4.4 Evaluation of Paraphrase Patterns

Using the method introduced in Section 3.4, we have generated 73,484 pairs of paraphrase patterns that appear at least two times in the corpus. We randomly selected 500 pairs and labeled them manually. The results show that the precision is 78.4%. Two examples are shown in Table 4, in which p_1 and p_2 are paraphrase patterns. Some slot fillers are also listed below. We realize that the method currently used for inducing paraphrase patterns is simple. Hence we will improve the method in our following experiments. Specifically, multiple slots will be allowed in a pair of patterns. In addition, we will try to apply the alignment techniques in the generation of paraphrase patterns, as Zhao et al. (2008) did.

p_1	[X] 文件 怎么 打开
p_2	如何 打开 [X] 文件
	(how to open [X] file)
slot	7z; ashx; aspx; bib; cda; cdfs; cmp; cpi; csf; csv; cur; dat; dek...
p_1	关于 [X] 的 诗词
p_2	有关 [X] 的 诗歌
	(poems about [X])
slot	草原 (prairies); 长江 (Yangtze River); 泰山 (Mount Tai); 乡愁 (nostalgia)...

Table 4: Examples of paraphrase patterns.

4.5 Analysis

Feature Contribution. To investigate the contributions of different features used in classification, we tried different feature combinations for each of our three classifiers. The results are shown in Table 5, in which "+" means the feature has contribution to the corresponding classifier. As can be seen, the character overlap rate feature (F_{COR}), cosine similarity feature (F_{CS}), and NE similarity

Feature	Q-T	Q-Q	T-T
F_F	+		
F_{LR}		+	
F_{WOR}			
F_{COR}	+	+	+
F_{CS}	+	+	+
F_{ED}		+	
F_{NE}	+	+	+
F_{PF}		+	

Table 5: Feature contribution.

feature (F_{NE}) are the most useful, which play important roles in all the three classifiers. The other features are useful in some of the classifiers except the word overlap rate feature (F_{WOR}). The classification results reported in prior sections are all achieved with the optimal feature combination.

Analysis of the Paraphrases. We combine the extracted paraphrase Q-T, Q-Q and T-T and get a total of 3,560,257 pairs of unique paraphrases. Statistics show that only 8380 pairs (0.24%) are from more than one source, which indicates that the intersection among the three sets is very small. Further statistics show that the average length of the queries and titles in the paraphrases is 6.69 (words).

To have a detailed analysis of the extracted paraphrases, we randomly selected 1000 pairs and manually labeled the precision, types, and domains. It is found that more than 43% of the paraphrases are paraphrase questions, in which *how* (36%), *what* (19%), and *yes/no* (14%) questions are the most common. In addition, we find that the precision of paraphrase questions (84.26%) is evidently higher than non-question paraphrases (65.14%). Those paraphrase questions are useful in question analysis and expansion in QA, which can hardly be extracted from other kinds of corpora.

As expected, the paraphrases we extract cover a variety of domains. However, around 50% of them are in the 7 most popular domains[7], including: (1) health and medicine, (2) documentary download, (3) entertainment, (4) software, (5) education and study, (6) computer game, (7) economy and finance. This analysis reflects what web users are most concerned about. These domains, especially (4) and (6), are not well covered by the parallel and comparable corpora previously used for paraphrase extraction.

[7]Note that pornographic queries have been filtered from the query log beforehand.

5 Conclusions and Future Directions

In this paper, we put forward a novel method that extracts paraphrases from search engine query logs. Our contribution is that we, for the first time, propose to extract paraphrases from user queries and the corresponding clicked document titles. Specifically, three kinds of paraphrases are extracted, which can be (1) a query and a hit title, (2) two queries that hit the same title, and (3) two titles hit by the same query. The extracted paraphrases are refined based on classification. Using the proposed method, we extracted over 3.5 million pairs of paraphrases from a query log of Baidu. Human evaluation results show that the precision of the paraphrases is above 70%. The results also show that we can generate high-quality paraphrase patterns from the extracted paraphrases.

Our future research will be conducted along the following directions. Firstly, we will use a much larger query log for paraphrase extraction, so as to enhance the coverage of paraphrases. Secondly, we plan to have a deeper study of the transitivity of paraphrasing. Simply speaking, we want to find out whether we can extract $\langle s_1, s_3 \rangle$ as paraphrases given that $\langle s_1, s_2 \rangle$ and $\langle s_2, s_3 \rangle$ are paraphrases.

6 Acknowledgments

We would like to thank Wanxiang Che, Hua Wu, and the anonymous reviewers for their useful comments on this paper.

References

Farooq Ahmad and Grzegorz Kondrak. 2005. Learning a Spelling Error Model from Search Query Logs. In *Proceedings of HLT/EMNLP*, pages 955-962.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with Bilingual Parallel Corpora. In *Proceedings of ACL*, pages 597-604.

Regina Barzilay and Lillian Lee. 2003. Learning to Paraphrase: An Unsupervised Approach Using Multiple-Sequence Alignment. In *Proceedings of HLT-NAACL*, pages 16-23.

Regina Barzilay and Kathleen R. McKeown. 2001. Extracting Paraphrases from a Parallel Corpus. In *Proceedings of ACL/EACL*, pages 50-57.

Rahul Bhagat and Deepak Ravichandran. 2008. Large Scale Acquisition of Paraphrases for Learning Surface Patterns. In *Proceedings of ACL-08: HLT*, pages 674-682.

Chris Callison-Burch, Philipp Koehn, and Miles Osborne. 2006. Improved Statistical Machine Translation Using Paraphrases. In *Proceedings of HLT-NAACL*, pages 17-24.

Chris Callison-Burch. 2008. Syntactic Constraints on Paraphrases Extracted from Parallel Corpora. In *Proceedings of EMNLP*, pages 196-205.

Hang Cui, Ji-Rong Wen, Jian-Yun Nie, Wei-Ying Ma. 2002. Probabilistic Query Expansion Using Query Logs In *Proceedings of WWW*, pages 325-332.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised Construction of Large Paraphrase Corpora: Exploiting Massively Parallel News Sources. In *Proceedings of COLING*, pages 350-356.

Pablo Ariel Duboue and Jennifer Chu-Carroll. 2006. Answering the Question You Wish They Had Asked: The Impact of Paraphrasing for Question Answering. In *Proceedings of HLT-NAACL*, pages 33-36.

Wei Gao, Cheng Niu, Jian-Yun Nie, Ming Zhou, Jian Hu, Kam-Fai Wong, and Hsiao-Wuen Hon. 2007. Cross-Lingual Query Suggestion Using Query Logs of Different Languages. In *Proceedings of SIGIR*, pages 463-470.

Ali Ibrahim, Boris Katz, Jimmy Lin. 2003. Extracting Structural Paraphrases from Aligned Monolingual Corpora. In *Proceedings of IWP*, pages 57-64.

Lidija Iordanskaja, Richard Kittredge, and Alain Polguère. 1991. Lexical Selection and Paraphrase in a Meaning-Text Generation Model. In Cécile L. Paris, William R. Swartout, and William C. Mann (Eds.): Natural Language Generation in Artificial Intelligence and Computational Linguistics, pages 293-312.

David Kauchak and Regina Barzilay. 2006. Paraphrasing for Automatic Evaluation. In *Proceedings of HLT-NAACL*, pages 455-462.

De-Kang Lin and Patrick Pantel. 2001. Discovery of Inference Rules for Question Answering. In *Natural Language Engineering* 7(4): 343-360.

Marius Paşca and Péter Dienes. 2005. Aligning Needles in a Haystack: Paraphrase Acquisition Across the Web. In *Proceedings of IJCNLP*, pages 119-130.

Marius Paşca. 2007. Weakly-supervised Discovery of Named Entities using Web Search Queries. In *Proceedings of CIKM*, pages 683-690.

Deepak Ravichandran and Eduard Hovy. 2002. Learning Surface Text Patterns for a Question Answering System. In *Proceedings of ACL*, pages 41-47.

Matthew Richardson. 2008. Learning about the World through Long-Term Query Logs. In *ACM Transactions on the Web* 2(4): 1-27.

Stefan Riezler, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu Mittal and Yi Liu. 2007. Statistical Machine Translation for Query Expansion in Answer Retrieval. In *Proceedings of ACL*, pages 464-471.

Satoshi Sekine and Hisami Suzuki. 2007. Acquiring Ontological Knowledge from Query Logs. In *Proceedings of WWW*, pages 1223-1224.

Yusuke Shinyama, Satoshi Sekine, and Kiyoshi Sudo. 2002. Automatic Paraphrase Acquisition from News Articles. In *Proceedings of HLT*, pages 40-46.

Ji-Rong Wen, Jian-Yun Nie, and Hong-Jiang Zhang. 2002. Query Clustering Using User Logs. In *ACM Transactions on Information Systems* 20(1): 59-81, 2002.

Shiqi Zhao, Haifeng Wang, Ting Liu, and Sheng Li. 2008. Pivot Approach for Extracting Paraphrase Patterns from Bilingual Corpora. In *Proceedings of ACL-08:HLT*, pages 780-788.

Shiqi Zhao, Ming Zhou, and Ting Liu. 2007. Learning Question Paraphrases for QA from Encarta Logs. In *Proceedings of IJCAI*, pages 1795-1800.

Leveraging Multiple MT Engines for Paraphrase Generation

Shiqi Zhao[†‡], **Haifeng Wang**[†], **Xiang Lan**[‡], **and Ting Liu**[‡]
[†]Baidu Inc.
[‡]HIT Center for Information Retrieval, Harbin Institute of Technology
{zhaoshiqi, wanghaifeng}@baidu.com,
{xlan, tliu}@ir.hit.edu.cn

Abstract

This paper proposes a method that leverages multiple machine translation (MT) engines for paraphrase generation (PG). The method includes two stages. Firstly, we use a multi-pivot approach to acquire a set of candidate paraphrases for a source sentence S. Then, we employ two kinds of techniques, namely the selection-based technique and the decoding-based technique, to produce a best paraphrase T for S using the candidates acquired in the first stage. Experimental results show that: (1) The multi-pivot approach is effective for obtaining plenty of valuable candidate paraphrases. (2) Both the selection-based and decoding-based techniques can make good use of the candidates and produce high-quality paraphrases. Moreover, these two techniques are complementary. (3) The proposed method outperforms a state-of-the-art paraphrase generation approach.

1 Introduction

This paper addresses the problem of paraphrase generation (PG), which seeks to generate paraphrases for sentences. PG is important in many natural language processing (NLP) applications. For example, in machine translation (MT), a sentence can be paraphrased so as to make it more translatable (Zhang and Yamamoto, 2002; Callison-Burch et al., 2006). In question answering (QA), a question can be paraphrased to improve the coverage of answer extraction (Duboue and Chu-Carroll, 2006; Riezler et al., 2007). In natural language generation (NLG), paraphrasing can help to increase the expressive power of the NLG systems (Iordanskaja et al., 1991).

In this paper, we propose a novel PG method. For an English sentence S, the method first acquires a set of candidate paraphrases with a multi-pivot approach, which uses MT engines to automatically translate S into multiple pivot languages and then translate them back into English. Furthermore, the method employs two kinds of techniques to produce a best paraphrase T for S using the candidates, i.e., the selection-based and decoding-based techniques. The former selects a best paraphrase from the candidates based on Minimum Bayes Risk (MBR), while the latter trains a MT model using the candidates and generates paraphrases with a MT decoder.

We evaluate our method on a set of 1182 English sentences. The results show that: (1) although the candidate paraphrases acquired by MT engines are noisy, they provide good raw materials for further paraphrase generation; (2) the selection-based technique is effective, which results in the best performance; (3) the decoding-based technique is promising, which can generate paraphrases that are different from the candidates; (4) both the selection-based and decoding-based techniques outperform a state-of-the-art approach SPG (Zhao et al., 2009).

2 Related Work

2.1 Methods for Paraphrase Generation

MT-based method is the mainstream method on PG. It regards PG as a monolingual machine translation problem, i.e., "translating" a sentence S into another sentence T in the same language.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1326–1334,
Beijing, August 2010

Quirk et al. (2004) first presented the MT-based method. They trained a statistical MT (SMT) model on a monolingual parallel corpus extracted from comparable news articles and applied the model to generate paraphrases. Their work shows that SMT techniques can be extended to PG. However, its usefulness is limited by the scarcity of monolingual parallel data.

To overcome the data sparseness problem, Zhao et al. (2008a) improved the MT-based PG method by training the paraphrase model using multiple resources, including monolingual parallel corpora, monolingual comparable corpora, bilingual parallel corpora, etc. Their results show that bilingual parallel corpora are the most useful among the exploited resources. Zhao et al. (2009) further improved the method by introducing a *usability sub-model* into the paraphrase model so as to generate varied paraphrases for different applications.

The main disadvantage of the MT-based method is that its performance heavily depends on the fine-grained paraphrases, such as paraphrase phrases and patterns, which provide paraphrase options in decoding. Hence one has to first extract fine-grained paraphrases from various corpora with different methods (Zhao et al., 2008a; Zhao et al., 2009), which is difficult and time-consuming.

In addition to the MT-based method, researchers have also investigated other methods for paraphrase generation, such as the pattern-based methods (Barzilay and Lee, 2003; Pang et al., 2003), thesaurus-based methods (Bolshakov and Gelbukh, 2004; Kauchak and Barzilay, 2006), and NLG-based methods (Kozlowski et al., 2003; Power and Scott, 2005).

2.2 Pivot Approach for Paraphrasing

Bannard and Callison-Burch (2005) introduced the pivot approach to extracting paraphrase phrases from bilingual parallel corpora. Their basic assumption is that two English phrases aligned with the same phrase in a foreign language (also called a pivot language) are potential paraphrases. Zhao et al. (2008b) extended the approach and used it to extract paraphrase patterns. Both of the above works have proved the effectiveness of the pivot approach in paraphrase extraction.

Pivot approach can also be used in paraphrase generation. It generates paraphrases by translating sentences from a source language to one (single-pivot) or more (multi-pivot) pivot languages and then translating them back to the source language. Duboue et al. (2006) first proposed the multi-pivot approach for paraphrase generation, which was specially designed for question expansion in QA. In addition, Max (2009) presented a single-pivot approach for generating sub-sentential paraphrases. A clear difference between our method and the above works is that we propose selection-based and decoding-based techniques to generate high-quality paraphrases using the candidates yielded from the pivot approach.

3 Multi-pivot Approach for Acquiring Candidate Paraphrases

A single-pivot PG approach paraphrases a sentence S by translating it into a pivot language PL with a MT engine MT_1 and then translating it back into the source language with MT_2. In this paper, a single-pivot PG system is represented as a triple (MT_1, PL, MT_2). A multi-pivot PG system is made up of a set of single-pivot systems with various pivot languages and MT engines. Given m pivot languages and n MT engines, we can build a multi-pivot PG system consisting of N ($N \leq n * m * n$) single-pivot ones, where $N = n * m * n$ iff all the n MT engines can perform bidirectional translation between the source and each pivot language.

In this work, we experiment with 6 pivot languages (Table 1) and 3 MT engines (Table 2) in the multi-pivot approach. All the 3 MT engines are off-the-shelf systems, in which Google and Microsoft translators are SMT engines, while Systran translator is a rule-based MT engine. Each MT engine can translate English to all the 6 pivot languages and back to English. We thereby construct a multi-pivot PG system consisting of 54 (3*6*3) single-pivot systems.

The advantages of the multi-pivot PG approach lie in two aspects. First, it effectively makes use of the vast bilingual data and translation rules underlying the MT engines. Second, the approach is simple, which just sends sentences to the online MT engines and gets the translations back.

Source Sentence	he said there will be major cuts in the salaries of high-level civil servants .
(GG, G, MS)	he said there **are significant** cuts in the salaries of high-level **officials** .
(GG, F, GG)	he said there will be **significant** cuts in the salaries of *top civil level* .
(MS, C, MS)	he said **that** there will be *a major senior civil service pay cut* .
(MS, F, ST)	he said there will be **great** cuts in the **wages** of the *high level civils servant* .
(ST, G, GG)	he said **that** there **are** major cuts in the salaries of **senior government officials** .

Table 3: Examples of candidate paraphrases obtained using the multi-pivot approach.

1	French (**F**)	4	Italian (**I**)
2	German (**G**)	5	Portuguese (**P**)
3	Spanish (**S**)	6	Chinese (**C**)

Table 1: Pivot languages used in the approach.

1	Google Translate (**GG**) (translate.google.com)
2	Microsoft Translator (**MS**) (www.microsofttranslator.com)
3	Systran Online Translation (**ST**) (www.systransoft.com)

Table 2: MT engines utilized in the approach.

4 Producing High-quality Paraphrases using the Candidates

Table 3 shows some examples of candidate paraphrases for a sentence. As can be seen, the candidates do provide some correct and useful paraphrase substitutes (in bold) for the source sentence. However, they also contain quite a few errors (in italic) due to the limited translation quality of the MT engines. The problem is even worse when the source sentences get longer and more complicated. Therefore, we need to combine the outputs of the multiple single-pivot PG systems and produce high-quality paraphrases out of them. To this end, we investigate two techniques, namely, the selection-based and decoding-based techniques.

4.1 Selection-based Technique

Given a source sentence S along with a set D of candidate paraphrases $\{T_1, T_2, ..., T_i, ...T_N\}$, the goal of the selection-based technique is to select the best paraphrase $\hat{T}_i$ for S from D. The paraphrase selection technique we propose is based on Minimum Bayes Risk (MBR). In detail, the MBR based technique first measures the quality of each candidate paraphrase $T_i \in D$ in terms of Bayes risk (BR), and then selects the one with the minimum BR as the best paraphrase. In detail, given S, a candidate $T_i \in D$, a reference paraphrase T[1], and a loss function $L(T, T_i)$ that measures the quality of T_i relative to T, we define the Bayes risk as follows:

$$BR(T_i) = E_{P(T,S)}[L(T, T_i)], \quad (1)$$

where the expectation is taken under the true distribution $P(T, S)$ of the paraphrases. According to (Bickel and Doksum, 1977), the candidate paraphrase that minimizes the Bayes risk can be found as follows:

$$\hat{T}_i = \arg\min_{T_i \in D} \sum_{T \in \mathcal{T}} L(T, T_i) P(T|S), \quad (2)$$

where $\mathcal{T}$ represents the space of reference paraphrases. In practice, however, the collection of reference paraphrases is not available. We thus construct a set $D' = D \bigcup \{S\}$ to approximate $\mathcal{T}$[2]. In addition, we cannot estimate $P(T|S)$ in Equation (2), either. Therefore, we make a simplification by assigning a constant c to $P(T|S)$ for each $T \in D'$, which can then be removed:

$$\hat{T}_i = \arg\min_{T_i \in D} \sum_{T \in D'} L(T, T_i). \quad (3)$$

Equation (3) can be further rewritten using a gain function $G(T, T_i)$ instead of the loss function:

[1]Here we assume that we have the collection of all possible paraphrases of S, which are used as references.

[2]The source sentence S is included in D' based on the consideration that a sentence is allowed to keep unchanged during paraphrasing.

$$\hat{T}_i = \arg\max_{T_i \in D} \sum_{T \in D'} G(T, T_i). \quad (4)$$

We define the gain function based on BLEU: $G(T, T_i) = BLEU(T, T_i)$. BLEU is a widely used metric in the automatic evaluation of MT (Papineni et al., 2002). It measures the similarity of two sentences by counting the overlapping n-grams (n=1,2,3,4 in our experiments):

$$BLEU(T, T_i) = BP \cdot \exp(\sum_{n=1}^{4} w_n \log p_n(T, T_i)),$$

where $p_n(T, T_i)$ is the n-gram precision of T_i and $w_n = 1/4$. BP (≤ 1) is a brevity penalty that penalizes T_i if it is shorter than T.

In summary, for each sentence S, the MBR based technique selects a paraphrase that is the most similar to all candidates and the source sentence. The underlying assumption is that correct paraphrase substitutes should be common among the candidates, while errors committed by the single-pivot PG systems should be all different. We denote this approach as **S-1** hereafter.

Approaches for comparison. In the experiments, we also design another two paraphrase selection approaches S-2 and S-3 for comparison with S-1.

S-2: S-2 selects the best single-pivot PG system from all the 54 ones. The selection is also based on MBR and BLEU. For each single-pivot PG system, we sum up its gain function values over a set of source sentences (i.e., $\sum_S \sum_{T_S \in D'_S} G(T_S, T_{Si})$). Then we select the one with the maximum gain value as the best single-pivot system. In our experiments, the selected best single-pivot PG system is (ST, P, GG), the candidate paraphrases acquired by which are then returned as the best paraphrases in S-2.

S-3: S-3 is a simple baseline, which just randomly selects a paraphrase from the 54 candidates for each source sentence S.

4.2 Decoding-based Technique

The selection-based technique introduced above has an inherent limitation that it can only select a paraphrase from the candidates. That is to say, it can never produce a perfect paraphrase if all the candidates have some tiny flaws. To solve this problem, we propose the decoding-based technique, which trains a MT model using the candidate paraphrases of each source sentence S and generates a new paraphrase T for S with a MT decoder.

major cuts	**high-level civil servants**
significant cuts	senior officials
major cuts*	high-level officials
important cuts	senior civil servants
big cuts	
great cuts	

Table 4: Extracted phrase pairs. (*This is called a *self-paraphrase* of the source phrase, which is generated when a phrase keeps unchanged in some of the candidate paraphrases.)

In this work, we implement the decoding-based technique using Giza++ (Och and Ney, 2000) and Moses (Hoang and Koehn, 2008), both of which are commonly used SMT tools. For a sentence S, we first construct a set of parallel sentences by pairing S with each of its candidate paraphrases: {(S,T_1),(S,T_2),...,(S,T_N)} ($N = 54$). We then run word alignment on the set using Giza++ and extract aligned phrase pairs as described in (Koehn, 2004). Here we only keep the phrase pairs that are aligned $\geq$3 times on the set, so as to filter errors brought by the noisy sentence pairs. The extracted phrase pairs are stored in a phrase table. Table 4 shows some extracted phrase pairs.

Note that Giza++ is sensitive to the data size. Hence it is interesting to examine if the alignment can be improved by augmenting the parallel sentence pairs. To this end, we have tried augmenting the parallel set for each sentence S by pairing any two candidate paraphrases. In this manner, C_N^2 sentence pairs are augmented for each S. We conduct word alignment using the $(N + C_N^2)$ sentence pairs and extract aligned phrases from the original N pairs. However, we have not found clear improvement after observing the results. Therefore, we do not adopt the augmentation strategy in our experiments.

Using the extracted phrasal paraphrases, we conduct decoding for the sentence S with Moses, which is based on a log-linear model. The default setting of Moses is used, except that the distortion model for phrase reordering is turned off[3]. The language model in Moses is trained using a 9 GB English corpus. We denote the above approach as **D-1** in what follows.

Approach for comparison. The main advantage of the decoding-based technique is that it allows us to customize the paraphrases for different requirements through tailoring the phrase table or tuning the model parameters. As a case study, this paper shows how to generate paraphrases with varied *paraphrase rates*[4].

D-2: The extracted phrasal paraphrases (including self-paraphrases) are stored in a phrase table, in which each phrase pair has 4 scores measuring their alignment confidence (Koehn et al., 2003). Our basic idea is to control the paraphrase rate by tuning the scores of the self-paraphrases. We thus extend D-1 to D-2, which assigns a weight λ ($\lambda > 0$) to the scores of the self-paraphrase pairs. Obviously, if we set $\lambda < 1$, the self-paraphrases will be penalized and the decoder will prefer to generate a paraphrase with more changes. If we set $\lambda > 1$, the decoder will tend to generate a paraphrase that is more similar to the source sentence. In our experiments, we set $\lambda = 0.1$ in D-2.

5 Experimental Setup

Our test sentences are extracted from the parallel reference translations of a Chinese-to-English MT evaluation[5], in which each Chinese sentence c has 4 English reference translations, namely e_1, e_2, e_3, and e_4. We use e_1 as a test sentence to paraphrase and e_2, e_3, e_4 as human paraphrases of e_1 for comparison with the automatically generated paraphrases. We process the test set by manually filtering ill-formed sentences, such as the ungrammatical or incomplete ones. 1182 out of 1357 test sentences are retained after filtering. Statistics show that about half of the test sentences are from news and the other half are from essays. The average length of the test sentences is 34.12 (words).

Score	Adequacy	Fluency
5	All	Flawless English
4	Most	Good English
3	Much	Non-native English
2	Little	Disfluent English
1	None	Incomprehensible

Table 5: Five point scale for human evaluation.

Manual evaluation is used in this work. A paraphrase T of a sentence S is manually scored based on a five point scale, which measures both the "adequacy" (i.e., how much of the meaning of S is preserved in T) and "fluency" of T (See Table 5). The five point scale used here is similar to that in the human evaluation of MT (Callison-Burch et al., 2007). In MT, adequacy and fluency are evaluated separately. However, we find that there is a high correlation between the two aspects, which makes it difficult to separate them. Thus we combine them in this paper.

We compare our method with a state-of-the-art approach SPG[6] (Zhao et al., 2009), which is a statistical approach specially designed for PG. The approach first collects a large volume of fine-grained paraphrase resources, including paraphrase phrases, patterns, and collocations, from various corpora using different methods. Then it generates paraphrases using these resources with a statistical model[7].

6 Experimental Results

We evaluate six approaches, i.e., S-1, S-2, S-3, D-1, D-2 and SPG, in the experiments. Each approach generates a 1-best paraphrase for a test sentence S. We randomize the order of the 6 paraphrases of each S to avoid bias of the raters.

[3]We conduct monotone decoding as previous work (Quirk et al., 2004; Zhao et al., 2008a, Zhao et al., 2009).

[4]The paraphrase rate reflects how different a paraphrase is from the source sentence.

[5]2008 NIST Open Machine Translation Evaluation: Chinese to English Task.

[6]SPG: Statistical Paraphrase Generation.

[7]We ran SPG under the setting of baseline-2 as described in (Zhao et al., 2009).

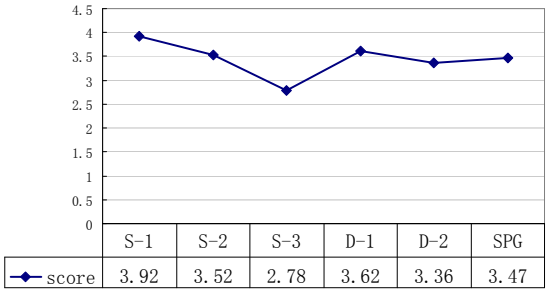


Figure 1: Evaluation results of the approaches.

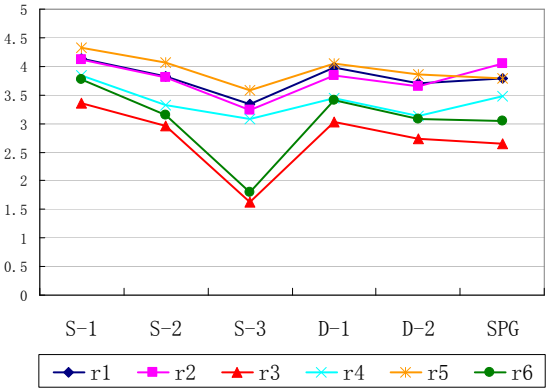


Figure 2: Evaluation results from each rater.

6.1 Human Evaluation Results

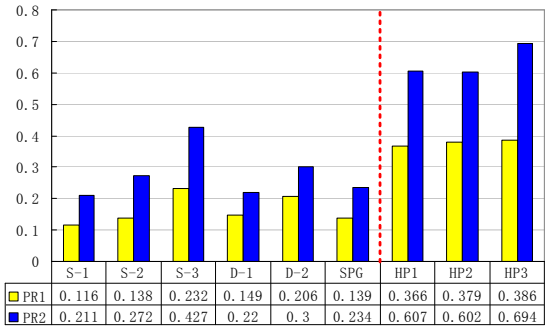
We have 6 raters in the evaluation, all of whom are postgraduate students. In particular, 3 raters major in English, while the other 3 major in computer science. Each rater scores the paraphrases of 1/6 test sentences, whose results are then combined to form the final scoring result. The average scores of the six approaches are shown in Figure 1. We can find that among the selection-based approaches, the performance of S-3 is the worst, which indicates that randomly selecting a paraphrase from the candidates works badly. S-2 performs much better than S-3, suggesting that the quality of the paraphrases acquired with the best single-pivot PG system are much higher than the randomly selected ones. S-1 performs the best in all the six approaches, which demonstrates the effectiveness of the MBR-based selection technique. Additionally, the fact that S-1 evidently outperforms S-2 suggests that it is necessary to extend a single-pivot approach to a multi-pivot one.

To get a deeper insight of S-1, we randomly sample 100 test sentences and manually score all of their candidates. We find that S-1 successfully picks out a paraphrase with the highest score for 72 test sentences. We further analyze the remaining 28 sentences for which S-1 fails and find that the failures are mainly due to the BLEU-based gain function. For example, S-1 sometimes selects paraphrases that have correct phrases but incorrect phrase orders, since BLEU is weak in evaluating phrase orders and sentence structures. In the next step we shall improve the gain function by investigating other features besides BLEU.

In the decoding-based approaches, D-1 ranks the second in the six approaches only behind S-1. We will further improve D-1 in the future rather than simply use Moses in decoding with the default setting. However, the value of D-1 lies in that it enables us to break down the candidates and generate new paraphrases flexibly. The performance decreases when we extend D-1 to D-2 to achieve a larger paraphrase rate. This is mainly because more errors are brought in when more parts of a sentence are paraphrased.

We can also find from Figure 1 that S-1, S-2, and D-1 all get higher scores than SPG, which shows that our method outperforms this state-of-the-art approach. This is more important if we consider that our method is lightweight, which makes no effort to collect fine-grained paraphrase resources beforehand. After observing the results, we believe that the outperformance of our method can be mainly ascribed to the selection-based and decoding-based techniques, since we avoid many errors by voting among the candidates. For instance, an ambiguous phrase may be incorrectly paraphrased by some of the single-pivot PG systems or the SPG approach. However, our method may obtain the correct paraphrase through statistics over all candidates and selecting the most credible one.

The human evaluation of paraphrases is subjective. Hence it is necessary to examine the coherence among the raters. The scoring results from the six raters are depicted in Figure 2. As it can be seen, they show similar trends though the raters have different degrees of strictness.



	S-1	S-2	S-3	D-1	D-2	SPG	HP1	HP2	HP3
PR1	0.116	0.138	0.232	0.149	0.206	0.139	0.366	0.379	0.386
PR2	0.211	0.272	0.427	0.22	0.3	0.234	0.607	0.602	0.694

Figure 3: Paraphrase rates of the approaches.

6.2 Paraphrase Rate

Human evaluation assesses the quality of paraphrases. However, the paraphrase rates cannot be reflected. A paraphrase that is totally transformed from the source sentence and another that is almost unchanged may get the same score. Therefore, we propose two strategies, i.e., PR1 and PR2, to compute the paraphrase rate:

$$PR1(T) = 1 - \frac{OL(S,T)}{L(S)}; \quad PR2(T) = \frac{ED(S,T)}{L(S)}.$$

Here, PR1 is defined based on word overlapping rate, in which $OL(S,T)$ denotes the number of overlapping words between a paraphrase T and its source sentence S, $L(S)$ denotes the number of words in S. PR2 is defined based on edit distance, in which $ED(S,T)$ denotes the edit distance between T and S. Obviously, PR1 only measures the percentage of words that are changed from S to T, whereas PR2 further takes word order changes into consideration. It should be noted that PR1 and PR2 not only count the correct changes between S and T, but also count the incorrect ones. We compute the paraphrase rate for each of the six approaches by averaging the paraphrase rates over the whole test set. The results are shown in the left part of Figure 3.

On the whole, the paraphrase rates of the approaches are not high. In particular, we can see that the paraphrase rate of D-2 is clearly higher than D-1, which is in line with our intention of designing D-2. We can also see that the paraphrase rate of S-3 is the highest among the approaches. We find it is mainly because the paraphrases generated with S-3 contain quite a lot of errors, which contribute most of the changes.

7 Analysis

7.1 Effectiveness of the Proposed Method

Our analysis starts from the candidate paraphrases acquired with the multi-pivot approach. Actually, the results of S-3 reflect the average quality of the candidate paraphrases. A score of 2.78 (See Figure 1) indicates that the candidates are unacceptable according to the human evaluation metrics. This is in line with our expectation that the automatically acquired paraphrases through a two-way translation are noisy. However, the results of S-1 and D-1 demonstrate that, using the selection-based and decoding-based techniques, we can produce paraphrases of good quality. Especially, S-1 gets a score of nearly 4, which suggests that the paraphrases are pretty good according to our metrics. Moreover, our method outperforms SPG built on pre-extracted fine-grained paraphrases. It shows that our method makes good use of the paraphrase knowledge from the large volume of bilingual data underlying the multiple MT engines.

7.2 How to Choose Pivot Languages and MT Engines in the Multi-pivot Approach

In our experiments, besides the six pivot languages used in the multi-pivot system, we have also tried another five pivot languages, including Arabic, Japanese, Korean, Russian, and Dutch. They are finally abandoned since we find that they perform badly. Our experience on choosing pivot languages is that: (1) a pivot language should be a language whose translation quality can be well guaranteed by the MT engines; (2) it is better to choose a pivot language similar to the source language (e.g., French - English), which is easier to translate; (3) the translation quality of a pivot language should not vary a lot among the MT engines. On the other hand, it is better to choose MT engines built on diverse models and corpora, which can provide different paraphrase options. We plan to employ a syntax-based MT engine in our further experiments besides the currently used phrase-based SMT and rule-based MT engines.

S	he said there will be major cuts in the salaries of high-level civil servants .
S-1	he said **that** there will be **significant** cuts in the salaries of **senior officials** .
S-2	he said there will be major cuts in salaries of *civil servants high level* .
S-3	he said **that** there will be **significant** cuts in the salaries of **senior officials** .
D-1	he said **,** there will be **significant** cuts in salaries of **senior** civil servants .
D-2	he said **,** there will be **significant** cuts in salaries of **senior officials** .
SPG	he said **that** there will be *the main* cuts in the **wages** of high-level civil servants .
HP1	he said there will be a **big salary cut for high-level government employees** .
HP2	he said **salaries of senior public servants would be slashed** .
HP3	he **claimed to implement huge salary cut to senior** civil servants .

Table 6: Comparing the automatically generated paraphrases with the human paraphrases.

7.3 Comparing the Selection-based and Decoding-based Techniques

It is necessary to compare the paraphrases generated via the selection-based and decoding-based techniques. As stated above, the selection-based technique can only select a paraphrase from the candidates, while the decoding-based technique can generate a paraphrase different from all candidates. In our experiments, we find that for about 90% test sentences, the paraphrases generated by the decoding-based approach D-1 are outside the candidates. In particular, we compare the paraphrases generated by S-1 and D-1 and find that, for about 40% test sentences, S-1 gets higher scores than D-1, while for another 21% test sentences, D-1 gets higher scores than S-1[8]. This indicates that the selection-based and decoding-based techniques are complementary. In addition, we find examples in which the decoding-based technique can generate a perfect paraphrase for the source sentence, even if all the candidate paraphrases have obvious errors. This also shows that the decoding-based technique is promising.

7.4 Comparing Automatically Generated Paraphrases with Human Paraphrases

We also analyze the characteristics of the generated paraphrases and compare them with the human paraphrases (i.e., the other 3 reference translations in the MT evaluation, see Section 5, which are denoted as HP1, HP2, and HP3). We find that, compared with the automatically generated paraphrases, the human paraphrases are more complicated, which involve not only phrase replacements, but also structure reformulations and even inferences. Their paraphrase rates are also much higher, which can be seen in the right part of Figure 3. We show the automatic and human paraphrases for the example sentence of this paper in Table 6. To narrow the gap between the automatic and human paraphrases, it is necessary to learn structural paraphrase knowledge from the candidates in the future work.

[8]For the rest 39%, S-1 and D-1 get identical scores.

8 Conclusions and Future Work

We put forward an effective method for paraphrase generation, which has the following contributions. First, it acquires a rich fund of paraphrase knowledge through the use of multiple MT engines and pivot languages. Second, it presents a MBR-based technique that effectively selects high-quality paraphrases from the noisy candidates. Third, it proposes a decoding-based technique, which can generate paraphrases that are different from the candidates. Experimental results show that the proposed method outperforms a state-of-the-art approach SPG.

In the future work, we plan to improve the selection-based and decoding-based techniques. We will try some standard system combination strategies, like confusion networks and consensus decoding. In addition, we will refine our evaluation metrics. In the current experiments, paraphrase correctness (adequacy and fluency) and paraphrase rate are evaluated separately, which seem to be incompatible. We plan to combine them together and propose a uniform metric.

References

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with Bilingual Parallel Corpora. In *Proceedings of ACL*, pages 597-604.

Regina Barzilay and Lillian Lee. 2003. Learning to Paraphrase: An Unsupervised Approach Using Multiple-Sequence Alignment. In *Proceedings of HLT-NAACL*, pages 16-23.

Peter J. Bickel and Kjell A. Doksum. 1977. *Mathematical Statistics: Basic Ideas and Selected Topics*. Holden-Day Inc., Oakland, CA, USA.

Igor A. Bolshakov and Alexander Gelbukh. 2004. Synonymous Paraphrasing Using WordNet and Internet. In *Proceedings of NLDB*, pages 312-323.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz and Josh Schroeder. 2007. (Meta-) Evaluation of Machine Translation. In *Proceedings of ACL-2007 Workshop on Statistical Machine Translation*, pages 136-158.

Chris Callison-Burch, Philipp Koehn, and Miles Osborne. 2006. Improved Statistical Machine Translation Using Paraphrases. In *Proceedings of HLT-NAACL*, pages 17-24.

Pablo Ariel Duboue and Jennifer Chu-Carroll. 2006. Answering the Question You Wish They Had Asked: The Impact of Paraphrasing for Question Answering. In *Proceedings of HLT-NAACL*, pages 33-36.

Hieu Hoang and Philipp Koehn. 2008. Design of the Moses Decoder for Statistical Machine Translation. In *Proceedings of ACL Workshop on Software engineering, testing, and quality assurance for NLP*, pages 58-65.

Lidija Iordanskaja, Richard Kittredge, and Alain Polguère. 1991. Lexical Selection and Paraphrase in a Meaning-Text Generation Model. In Cécile L. Paris, William R. Swartout, and William C. Mann (Eds.): Natural Language Generation in Artificial Intelligence and Computational Linguistics, pages 293-312.

David Kauchak and Regina Barzilay. 2006. Paraphrasing for Automatic Evaluation. In *Proceedings of HLT-NAACL*, pages 455-462.

Philipp Koehn. 2004. Pharaoh: a Beam Search Decoder for Phrase-Based Statistical Machine Translation Models: User Manual and Description for Version 1.2.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of HLT-NAACL*, pages 127-133.

Raymond Kozlowski, Kathleen F. McCoy, and K. Vijay-Shanker. 2003. Generation of single-sentence paraphrases from predicate/argument structure using lexico-grammatical resources. In *Proceedings of IWP*, pages 1-8.

Aurélien Max. 2009. Sub-sentential Paraphrasing by Contextual Pivot Translation. In *Proceedings of the 2009 Workshop on Applied Textual Inference, ACL-IJCNLP 2009*, pages 18-26.

Franz Josef Och and Hermann Ney. 2000. Improved Statistical Alignment Models. In *Proceedings of ACL*, pages 440-447.

Bo Pang, Kevin Knight, and Daniel Marcu. 2003. Syntax-based Alignment of Multiple Translations: Extracting Paraphrases and Generating New Sentences. In *Proceedings of HLT-NAACL*, pages 102-109.

Kishore Papineni, Salim Roukos, Todd Ward, Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of ACL*, pages 311-318.

Richard Power and Donia Scott. 2005. Automatic generation of large-scale paraphrases. In *Proceedings of IWP*, pages 73-79.

Chris Quirk, Chris Brockett, and William Dolan. 2004. Monolingual Machine Translation for Paraphrase Generation. In *Proceedings of EMNLP*, pages 142-149.

Stefan Riezler, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu Mittal and Yi Liu. 2007. Statistical Machine Translation for Query Expansion in Answer Retrieval. In *Proceedings of ACL*, pages 464-471.

Yujie Zhang and Kazuhide Yamamoto. 2002. Paraphrasing of Chinese Utterances. In *Proceedings of COLING*, pages 1163-1169.

Shiqi Zhao, Xiang Lan, Ting Liu, and Sheng Li. 2009. Application-driven Statistical Paraphrase Generation. In *Proceedings of ACL-IJCNLP 2009*, pages 834-842.

Shiqi Zhao, Cheng Niu, Ming Zhou, Ting Liu, and Sheng Li. 2008a. Combining Multiple Resources to Improve SMT-based Paraphrasing Model. In *Proceedings of ACL-08:HLT*, pages 1021-1029.

Shiqi Zhao, Haifeng Wang, Ting Liu, and Sheng Li. 2008b. Pivot Approach for Extracting Paraphrase Patterns from Bilingual Corpora. In *Proceedings of ACL-08:HLT*, pages 780-788.

Resolving Surface Forms to Wikipedia Topics

Yiping Zhou Lan Nie Omid Rouhani-Kalleh Flavian Vasile Scott Gaffney
Yahoo! Labs at Sunnyvale
{zhouy,lannie,omid,flavian,gaffney}@yahoo-inc.com

Abstract

Ambiguity of entity mentions and concept references is a challenge to mining text beyond surface-level keywords. We describe an effective method of disambiguating surface forms and resolving them to Wikipedia entities and concepts. Our method employs an extensive set of features mined from Wikipedia and other large data sources, and combines the features using a machine learning approach with automatically generated training data. Based on a manually labeled evaluation set containing over 1000 news articles, our resolution model has 85% precision and 87.8% recall. The performance is significantly better than three baselines based on traditional context similarities or sense commonness measurements. Our method can be applied to other languages and scales well to new entities and concepts.

1 Introduction

Ambiguity in natural language is prevalent and, as such, it can be a difficult challenge for information retrieval systems and other text mining applications. For example, a search for "*Ford*" in Yahoo! News retrieves about 40 thousand articles containing *Ford* referring to a company (Ford Motors), an athlete (Tommy Ford), a place (Ford City), etc. Due to reference ambiguity, even if we knew the user was only interested in the company, they would still have to contend with articles referring to the other concepts as well.

In this paper we focus on the problem of resolving references of named-entities and concepts in natural language through their textual *surface forms*. Specifically, we present a method of resolving surface forms in general text documents to Wikipedia entries. The tasks of resolution and disambiguation are nearly identical; we make the distinction that resolution specifically applies when a known set of referent concepts are given a priori. Our approach differs from others in multiple aspects including the following.

1) We employ a rich set of disambiguation features leveraging mining results from large-scale data sources. We calculate context-sensitive features by extensively mining the categories, links and contents of the entire Wikipedia corpus. Additionally we make use of context-independent data mined from various data sources including Web user-behavioral data and Wikipedia. Our features also capture the one-to-one relationship between a surface form and its referent.

2) We use machine learning methods to train resolution models with a large automatically labeled training set. Both ranking-based and classification-based resolution approaches are explored.

3) Our method disambiguates both entities and word senses. It scales well to new entities and concepts, and it can be easily applied to other languages.

We propose an extensive set of metrics to evaluate not only overall resolution performance but also out-of-Wikipedia prediction. Our systems for English language are evaluated using real-world test sets and compared with a number of baselines. Evaluation results show that our systems consistently and significantly outperform others across all test sets.

The paper is organized as follows. We first describe related research in Section 2, followed by an introduction of Wikipedia in Section 3. We then introduce our learning method in Section 4 and our features in Section 5. We show our experimental results in Section 6, and finally close with a discussion of future work.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1335–1343,
Beijing, August 2010

2 Related Work

Named entity disambiguation research can be divided into two categories: some works (Bagga and Baldwin, 1998; Mann and Yarowsky, 2003; Pedersen et al., 2005; Fleischman and Hovy, 2004; Ravin and Kazi, 1999) aim to cluster ambiguous surface forms to different groups, with each representing a unique entity; others (Cucerzan, 2007; Bunescu and Paşca, 2006; Han and Zhao, 2009; Milne and Witten, 2008a; Milne and Witten, 2008b) resolve a surface form to an entity or concept extracted from existing knowledge bases. Our work falls into the second category.

Looking specifically at resolution, Bunescu and Pasca (2006) built a taxonomy SVM kernel to enrich a surface form's representation with words from Wikipedia articles in the same category. Cucerzan (2007) employed context vectors consisting of phrases and categories extracted from Wikipedia. The system also attempted to disambiguate all surface forms in a context simultaneously, with the constraint that their resolved entities should be globally consistent on the category level as much as possible. Milne and Witten (2008a, 2008b) proposed to use Wikipedia's link structure to capture the *relatedness* between Wikipedia entities so that a surface form is resolved to an entity based on its relatedness to the surface form's surrounding entities. Besides relatedness, they also define a *commonness* feature that captures how common it is that a surface form links to a particular entity in general. Han and Zhao (2009) defined a novel alignment strategy to calculate similarity between surface forms based on semantic relatedness in the context.

Milne and Witten's work is most related to what we propose here in that we also employ features similar to their relatedness and commonness features. However, we add to this a much richer set of features which are extracted from Web-scale data sources beyond Wikipedia, and we develop a machine learning approach to automatically blend our features using completely automatically generated training data.

3 Wikipedia

Wikipedia has more than 200 language editions, and the English edition has more than 3 million articles as of March 2009. Newsworthy events are often added to Wikipedia within days of occurrence; Wikipedia has bi-weekly snapshots available for download.

Each article in Wikipedia is uniquely identified by its title which is usually the most common surface form of an entity or concept. Each article includes body text, outgoing links and categories. Here is a sample sentence in the article titled "Aristotle" in wikitext format. "*Together with Plato and [[Socrates]] (Plato's teacher), Aristotle is one of the most important founding figures in [[Western philosophy]]*." Near the end of the article, there are category links such as "*[[Category:Ancient Greek mathematicians]]*". The double brackets annotate outgoing links to other Wikipedia articles with the specified titles. The category names are created by authors. Articles and category names have many-to-many relationships.

In addition to normal articles, Wikipedia also has special types of articles such as redirect articles and disambiguation articles. A redirect article's title is an alternative surface form for a Wikipedia entry. A disambiguation article lists links to similarly named articles, and usually its title is a commonly used surface form for multiple entities and concepts.

4 Method of Learning

Our goal is to resolve surface forms to entities or concepts described in Wikipedia. To this end, we first need a recognizer to detect surface forms to be resolved. Then we need a resolver to map a surface form to the most probable entry in Wikipedia (or to *out-of-wiki*) based on the context.

Recognizer: We first create a set of Wikipedia (article) entries $E = \{e_1, e_2, ...\}$ to which we want to resolve surface forms. Each entry's surface forms are mined from multiple data sources. Then we use simple string match to recognize surface forms from text documents.

Among all Wikipedia entries, we exclude those with low importance. In our experiments, we removed the entries that would not interest general Web users, such as stop words and punctuations. Second, we collect surface forms for entries in E using Wikipedia and Web search query click logs based on the following assumptions:

- Each Wikipedia article title is a surface form for the entry. Redirect titles are taken as alternative surface forms for the target entry.
- The anchor text of a link from one article to another is taken as an alternative surface form for the linked-to entry.
- Web search engine queries resulting in user clicks on a Wikipedia article are taken as alternative surface forms for the entry.

As a result, we get a number of surface forms for each entry e_i. If we let s_{ij} denote the j-th surface form for entry i, then we can represent our entry dictionary as $EntSfDict = \{<e_1, (s_{11}, s_{12}, ...)>, <e_2, (s_{21}, s_{22}, ...)>, ...\}$.

Resolver: We first build a labeled training set automatically, and then use supervised learning methods to learn models to resolve among Wikipedia entries. In the rest of this section we describe the resolver in details.

4.1 Automatically Labeled Data

To learn accurate models, supervised learning methods require training data with both large quantity and high quality, which often takes lots of human labeling effort. However, in Wikipedia, links provide a *supervised* mapping from surface forms to article entries. We use these links to automatically generate training data. If a link's anchor text is a surface form in *EntSfDict*, we extract the anchor text as surface form *s* and the link's destination article as Wikipedia entry *e*, then add the pair *(s, e)* with a positive judgment to our labeled example set. Continuing, we use *EntSfDict* to find other Wikipedia entries for which *s* is a surface form and create negative examples for these and add them to our labeled example set. If *e* does not exist in *EntSfDict* (for example, if the link points to a Wikipedia article about a stop word), then a negative training example is created for every Wikipedia entry to which *s* may resolve. We use *oow* (out-of-wiki) to denote this case.

Instead of article level coreference resolution, we only match partial names with full names based on the observation that surface forms for named entities are usually capitalized word sequences in English language and a named entity is often mentioned by a long surface form followed by mentions of short forms in the same article. For each pair *(s, e)* in the labeled example set, if *s* is a partial name of a full name *s'* occurring earlier in the same document, we replace *(s, e)* with *(s', e)* in the labeled example set.

Using this methodology we created 2.4 million labeled examples from only 1% of English Wikipedia articles. The abundance of data made it possible for us to experiment on the impact of training set size on model accuracy.

4.2 Learning Algorithms

In our experiments we explored both Gradient Boosted Decision Trees (GBDT) and Gradient Boosted Ranking (GBRank) to learn resolution models. They both can easily combine features of different scale and with missing values. Other supervised learning methods are to be explored in the future.

GBDT: We use the stochastic variant of GBDTs (Friedman, 2001) to learn a binary logistic regression model with the judgments as the target. GBDTs compute a function approximation by performing a numerical optimization in the function space. It is done in multiple stages, with each stage modeling residuals from the model of the last stage using a small decision tree. A brief summary is given in Algorithm 1. In the stochastic version of GBDT, one sub-samples the training data instead of using the entire training set to compute the loss function.

Algorithm 1 GBDTs

Input: training data $\{(x_i, y_i)\}_{i=1}^{N}$, loss function $L[y, f(x)]$, the number of nodes for each tree J, the number of trees M.

1: Initialize $f(x)=f^0$
2: For $m = 1$ to M
2.1: For $i = 1$ to N, compute the negative gradient by taking the derivative of the loss with respect to $f(x)$ and substitute with y_i and $f_i^{m-1}(x_i)$.
2.2: Fit a J-node regression tree to the components of the negative gradient.
2.3: Find the within-node updates a_j^m for $j = 1$ to J by performing J univariate optimizations of the node contributions to the estimated loss.
2.4: Do the update $f_i^m(x_i) = f_i^{m-1}(x_i) + r \times a_j^m$, where j is the node that x_i belongs to, r is learning rate.
3: End for
4: Return f^M

In our setting, the loss function is a negative binomial log-likelihood, x_i is the feature vector for a surface-form and Wikipedia-entry pair (s_i, e_i), and y_i is +1 for positive judgments and -1 is for negative judgments.

GBRank: From a given surface form's judgments we can infer that the correct Wikipedia entry is preferred over other entries. This allows us to derive pair-wise preference judgments from absolute judgments and train a model to rank all the Wikipedia candidate entries for each surface form. Let $S = \{(x_i, x_i^{'}) \mid l(x_i) \geq l(x_i^{'}), i = 1,...,N\}$ be the set of preference judgments, where x_i and x_i' are the feature vectors for two pairs of surface-forms and Wikipedia-entry, $l(x_i)$ and $l(x_i')$ are their absolute judgments respectively. GBRank (Zheng et al., 2007) tries to learn a function h such that $h(x_i) \geq h(x_i^{'})$ for $(x_i, x_i^{'}) \in S$. A sketch of the algorithm is given in Algorithm 2.

Algorithm 2 GBRank

1: Initialize $h=h_0$
2: For $k=1$ to K
2.1: Use h_{k-1} as an approximation of h and compute
$S^{+} = \{(x_i, x_i^{'}) \in S \mid h_{k-1}(x_i) \geq h_{k-1}(x_i^{'}) + \tau\}$
$S^{-} = \{(x_i, x_i^{'}) \in S \mid h_{k-1}(x_i) < h_{k-1}(x_i^{'}) + \tau\}$
where $\tau = \alpha(l(x_i) - l(x_i^{'}))$
2.2: Fit a regression function g_k using GBDT and the incorrectly predicted examples
$\{(x_i, h_{k-1}(x_i^{'}) + \tau), (x_i^{'}, h_{k-1}(x_i) - \tau) \mid (x_i, x_i^{'}) \in S^{-}\}$
2.3: Do the update
$h_k(x) = (kh_{k-1}(x) + \eta g_k(x))/(k+1)$, where η is learning rate.
3: End for
4: Return h_K

We use a tuning set independent from the training set to select the optimal parameters for GBDT and GBRank. This includes the number of trees M, the number of nodes J, the learning rate r, and the sampling rate for GBDT; and for GBRank we select K, α and η.

The feature importance measurement given by GBDT and GBRank is computed by keeping track of the reduction in the loss function at each feature variable split and then computing the total reduction of loss along each explanatory feature variable. We use it to analyze feature effectiveness.

4.3 Prediction

After applying a resolution model on the given test data, we obtain a score for each surface-form and Wikipedia-entry pair *(s, e)*. Among all the pairs containing *s*, we find the pair with the highest score, denoted by $(s, \tilde{e})$.

It's very common that a surface form refers to an entity or concept not defined in Wikipedia. So it's important to correctly predict whether the given surface form cannot be mapped to any Wikipedia entry in *EntSfDict*.

We apply a threshold to the scores from resolution models. If the score for $(s, \tilde{e})$ is lower than the threshold, then the prediction is *oow* (see Section 4.1), otherwise $\tilde{e}$ is predicted to be the entry referred by *s*. We select thresholds based on F1 (see Section 6.2) on a tuning set that is independent from our training set and test set.

5 Features

For each surface-form and Wikipedia-entry pair *(s, e)*, we create a feature vector including features capturing the context surrounding *s* and features independent of the context. They are context-dependent and context-independent features respectively. Various data sources are mined to extract these features, including Wikipedia articles, Web search query-click logs, and Web-user browsing logs. In addition, *(s, e)* is compared to all pairs containing *s* based on above features and the derived features are called differentiation features.

5.1 Context-dependent Features

These features measure whether the given surface form *s* resolving to the given Wikipedia entry *e* would make the given document more coherent. They are based on 1) the vector representation of *e*, and 2) the vector representation of the context of *s* in a document *d*.

Representation of *e*: By thoroughly mining Wikipedia and other large data sources we extract contextual clues for each Wikipedia entry *e* and formulate its representation in the following ways.

1) *Background representation*. The overall background description of *e* is given in the corresponding Wikipedia article, denoted as A_e. Naturally, a bag of terms and surface forms in A_e can represent *e*. So we represent *e* by a back-

ground word vector E_{bw} and a background surface form vector E_{bs}, in which each element is the occurrence count of a word or a surface form in A_e's first paragraph.

2) *Co-occurrence representation.* The terms and surface forms frequently co-occurring with *e* capture its contextual characteristics. We first identify all the Wikipedia articles linking to A_e. Then, for each link pointing to A_e we extract the surrounding words and surface forms within a window centered on the anchor text. The window size is set to 10 words in our experiment. Finally, we select the words and surface forms with the top co-occurrence frequency, and represent *e* by a co-occurring word vector E_{cw} and a co-occurring surface form vector E_{cs}, in which each element is the co-occurrence frequency of a selected word or surface form.

3) *Relatedness representation*. We analyzed the relatedness between Wikipedia entries from different data sources using various measurements, and we computed over 20 types of relatedness scores in our experiments. In the following we discuss three types as examples. The first type is computed based on the overlap between two Wikipedia entries' categories. The second type is mined from Wikipedia inter-article links. (In our experiments, two Wikipedia entries are considered to be related if the two articles are mutually linked to each other or co-cited by many Wikipedia articles.) The third type is mined from Web-user browsing data based on the assumption that two Wikipedia articles co-occurring in the same browsing session are related. We used approximately one year of Yahoo! user data in our experiments. A number of different metrics are used to measure the relatedness. For example, we apply the algorithm of Google distance (Milne and Witten, 2008b) on Wikipedia links to calculate the Wikipedia link-based relatedness, and use mutual information for the browsing-session-based relatedness. In summary, we represent *e* by a related entry vector E_r for each type of relatedness, in which each element is the relatedness score between *e* and a related entry.

Representation of *s*: We represent a surface form's context as a vector, then calculate a context-dependent feature for a pair $<s,e>$ by a similarity function *Sim* from two vectors. Here are examples of context representation.

1) *s* is represented by a word vector S_w and a surface form vector S_s, in which each element is the occurrence count of a word or a surface form surrounding *s*. We calculate each vector's similarity with the background and co-occurrence representation of *e*, and it results in $Sim(S_w, E_{bw})$, $Sim(S_w, E_{cw})$, $Sim(S_s, E_{bs})$ and $Sim(S_s, E_{cs})$.

2) *s* is represented by a Wikipedia entry vector S_e, in which each element is a Wikipedia entry to which a surrounding surface form *s* could resolve. We calculate its similarity with the relatedness representation of *e*, and it results in $Sim(S_e, E_r)$.

In the above description, similarity is calculated by dot product or in a summation-of-maximum fashion. In our experiments we extracted surrounding words and surface forms for *s* from the whole document or from the text window of 55 tokens centered on *s*, which resulted in 2 sets of features. We created around 50 context-dependent features in total.

5.2 Context-independent Features

These features are extracted from data beyond the document containing s. Here are examples.

- During the process of building the dictionary *EntSfDict* as described in Section 4, we count how often *s* maps to *e* and estimate the probability of *s* mapping to *e* for each data source. These are the commonness features.
- The number of Wikipedia entries that *s* could map to is a feature about the ambiguity of *s*.
- The string similarity between *s* and the title of A_e is used as a feature. In our experiments string similarity was based on word overlap.

5.3 Differentiation Features

Among all surface-form and Wikipedia-entry pairs that contain *s*, at most one pair gets the positive judgment. Based on this observation we created differentiation features to represent how *(s, e)* is compared to other pairs for *s*. They are derived from the context-dependent and context-independent features described above. For example, we compute the difference between the string similarity for *(s, e)* and the maximum string similarity for all pairs containing *s*. The derived feature value would be zero if *(s, e)* has larger string similarity than other pairs containing *s*.

6 Experimental Results

In our experiments we used the Wikipedia snapshot for March 6th, 2009. Our dictionary *EntSfDict* contains 3.5 million Wikipedia entries and 6.5 million surface forms.

A training set was created from randomly selected Wikipedia articles using the process described in Section 4.1. We varied the number of Wikipedia articles from 500 to 40,000, but the performance did not increase much after 5000. The experimental results reported in this paper are based on the training set generated from 5000 articles. It contains around 1.4 million training examples. There are approximately 300,000 surface forms, out of which 28,000 are the *oow* case.

Around 400 features were created in total, and 200 of them were selected by GBDT and GBRank to be used in our resolution models.

6.1 Evaluation Datasets

Three datasets from different data sources are used in evaluation.

1) *Wikipedia hold-out set*. Using the same process for generating training data and excluding the surface forms appearing in the training data, we built the hold-out set from approximately 15,000 Wikipedia articles, containing around 600,000 labeled instances. There are 400,000 surface forms, out of which 46,000 do not resolve to any Wikipedia entry.

2) *MSNBC News test set*. This entity disambiguation data set was introduced by Cucerzan (2007). It contains 200 news articles collected from ten MSNBC news categories as of January 2, 2007. Surface forms were manually identified and mapped to Wikipedia entities. The data set contains 756 surface forms. Only 589 of them are contained in our dictionary *EntSfDict*, mainly because *EntSfDict* excludes surface forms of out-of-Wikipedia entities and concepts. Since the evaluation task is focused on resolution performance rather than recognition, we exclude the missing surface forms from the labeled example set. The final dataset contains 4,151 labeled instances. There are 589 surface forms and 40 of them do not resolve to any Wikipedia entry.

3) *Yahoo! News set*. One limitation of the MSNBC test set is the small size. We built a much larger data set by randomly sampling around 1,000 news articles from Yahoo! News over 2008 and had them manually annotated. The experts first identified *person*, *location* and *organization* names, then mapped each name to a Wikipedia article if the article is about the entity referred to by the name. We didn't include more general concepts in this data set to make the manual effort easier. This data set contains around 100,000 labeled instances. The data set includes 15,387 surface forms and 3,532 of them cannot be resolved to any Wikipedia entity. We randomly split the data set to 2 parts of equal size. One part is used to tune parameters of GBDT and GBRank and select thresholds based on F1 value. The evaluation results presented in this paper is based on the remaining part of the Yahoo! News set.

6.2 Metrics

The possible outcomes from comparing a resolution system's prediction with ground truth can be categorized into the following types.

- True Positive (TP), the predicted *e* was correctly referred to by *s*.
- True Negative (TN), *s* was correctly predicted as resolving to *oow*.
- Mismatch (MM), the predicted *e* was not correctly referred to by *s* and should have been *e'* from *EntSfDict*.
- False Positive (FP), the predicted *e* was not correctly referred to by *s* and should have been *oow*.
- False Negative (FN), the predicted *oow* is not correct and should have been *e'* from *EntSfDict*.

Similar to the widely used metrics for classification systems, we use following metrics to evaluate disambiguation performance.

$$precision = \frac{TP}{TP + FP + MM} \qquad recall = \frac{TP}{TP + FN + MM}$$

$$F1 = \frac{2 \times precision \times recall}{precision + recall} \qquad accuracy = \frac{TP + TN}{TP + FP + TN + FN + MM}$$

In the Yahoo! News test set, 23.5% of the surface forms do not resolve to any Wikipedia entries, and in the other two test sets the percentages of *oow* are between 10% and 20%. This demonstrates it is necessary in real-world applications to explicitly measure *oow* prediction. We propose following metrics.

$$precision\_oow = \frac{TN}{TN + FN} \qquad recall\_oow = \frac{TN}{TN + FP}$$

$$F1\_oow = \frac{2 \times precision\_oow \times recall\_oow}{precision\_oow + recall\_oow}$$

6.3 Evaluation Results

With our training set we trained one resolution model using GBDT (named as *WikiRes-c*) and another resolution model using GBRank (named as *WikiRes-r*). The models were evaluated along with the following systems.

1) *Baseline-r*: each surface form *s* is randomly mapped to *oow* or a candidate entry for *s* in *EntSfDict*.

2) *Baseline-p*: each surface form *s* is mapped to the candidate entry *e* for *s* with the highest commonness score. The commonness score is linear combination of the probability of *s* being mapped to *e* estimated from different data sources. The commonness score is among the features used in *WikiRes-c* and *WikiRes-r*.

3) *Baseline-m*: we implemented the approach brought by Cucerzan (2007) based on our best understanding. Since we use a different version of Wikipedia and a different entity recognition approach, the evaluation result differs from the result presented in their paper. But we believe our implementation follows the algorithm described in their paper.

In Table 1 we present the performance for each system on the Yahoo! News test set and the MSNBC test set. The performance of *WikiRes-c* and *WikiRes-r* are computed after we apply the thresholds selected on the tuning set described in Section 6.1. In the upper half of Table 1, the three baselines use the thresholds that lead to the best F1 on the Yahoo! News test set. In the lower half of Table 1, the three baselines use the thresholds that lead to the best F1 on the MSNBC test set.

Among the three baselines, *Baseline-r* has the lowest performance. *Baseline-m* uses a few context-sensitive features and *Baseline-p* uses a context-independent feature. These two types of features are both useful, but *Baseline-p* shows better performance, probably because the surface forms in our test sets are dominated by common senses. In our resolution models, these features are combined together with many other features calculated from different large-scale data sources and on different granularity levels. As shown in Table 1, both of our resolution solutions substantially outperform other systems. Furthermore, *WikiRes-c* and *WikiRes-r* have similar performance.

	Precision	Recall	F1	Accuracy	p-value
Yahoo! News Test Set					
Baseline-r	47.023	60.831	53.043	47.023	0
Baseline-p	73.869	88.157	80.383	73.175	5.2e-78
Baseline-m	62.240	80.517	70.208	62.240	1.3e-160
WikiRes-r	83.406	**88.858**	86.046	80.717	0.012
WikiRes-c	**85.038**	87.831	**86.412**	**81.463**	---
MSNBC Test Set					
Baseline-r	60.272	64.545	62.335	60.272	8.9e-19
Baseline-p	82.292	86.182	84.192	82.003	0.306
Baseline-m	78.947	84.545	81.651	78.947	0.05
WikiRes-r	**88.785**	**86.364**	**87.558**	**84.550**	0.102
WikiRes-c	88.658	85.273	86.932	83.192	---

Table 1. Performance on the Yahoo! News Test Set and the MSNBC Test set

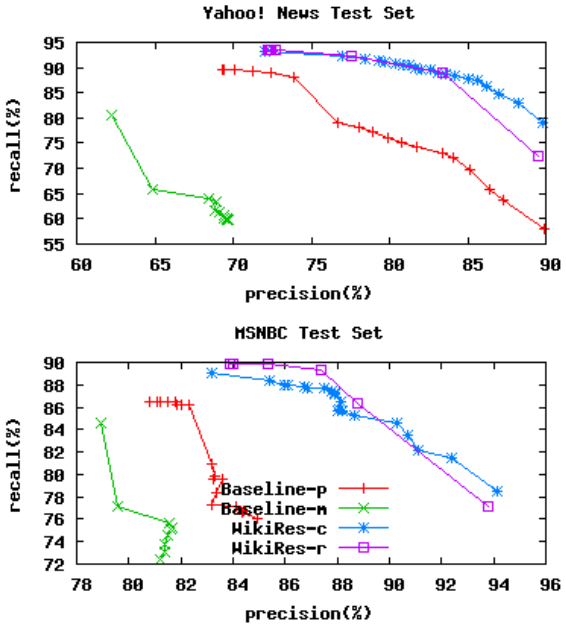


Figure 1. Precision-recall on the Yahoo! News Test Set and the MSNBC Test Set

We compared *WikiRes-c* with each competitor and from the statistical significance test results in the last column of Table 1 we see that on the Yahoo! News test set *WikiRes-c* significantly outperforms others. The p-values for the MSNBC test set are much higher than for the Yahoo! News test set because the MSNBC test set is much smaller.

Attempting to address this point, we see that the F1 values of *WikiRes* on the MSNBC test set and on the Yahoo! News test set only differs by a couple percentage points, although, these test sets were created independently. This suggests the objectivity of our method for creating the Yahoo! News test set and provides a way to measure resolution model performance on what

would occur in a general news corpus in a statistically significant manner.

In Figure 1 we present the precision-recall curves on the Yahoo! News and the MSNBC test sets. We see that our resolution models are substantially better than the other two baselines at any particular precision or recall value on both test sets. *Baseline-r* is not included in the comparison since it does not have the tradeoff between precision and recall. We find the precision-recall curve of *WikiRes-r* is very similar to *WikiRes-c* at the lower precision area, but its recall is much lower than other systems after precision reaches around 90%. So, in Figure 1 the curves of *WikiRes-r* are truncated at the high precision area.

In Table 2 we compare the performance of out-of-Wikipedia prediction. The comparison is done on the Yahoo! News test set only, since there are only 40 surface forms of *oow* case in the MSNBC test set. Each system's threshold is the same as that used for the upper half of Table 1. The results show our models have substantially higher precision and recall than *Baseline-p* and *Baseline-m*. From the statistical significance test results in the last column, we can see that *WikiRes-c* significantly outperforms *Baseline-p* and *Baseline-m*. Also, our current approaches still have room to improve in the area of out-of-Wikipedia prediction.

We also evaluated our models on a Wikipedia hold-out set. The model performance is greater than that obtained from the previous two test sets because the hold-out set is more similar to the training data source itself. Again, our models perform better than others.

From the feature importance lists of our GBDT model and GBRank model, we find that the commonness features, the features based on Wikipedia entries' co-occurrence representation and the corresponding differentiation features are the most important.

	Precision	Recall	F1	p-value
Baseline-p	64.907	22.152	33.03	1.6e-20
Baseline-m	47.207	44.78	45.961	1.3e-34
WikiRes-r	**68.166**	52.994	59.630	0.084
WikiRes-c	67.303	**59.777**	**63.317**	---

Table 2. Performance of Out-of-Wikipedia Prediction on the Yahoo! News Test Set

7 Conclusions

We have described a method of learning to resolve surface forms to Wikipedia entries. Using this method we can enrich the unstructured documents with structured knowledge from Wikipedia, the largest knowledge base in existence. The enrichment makes it possible to represent a document as a machine-readable network of senses instead of just a bag of words. This can supply critical semantic information useful for next-generation information retrieval systems and other text mining applications.

Our resolution models use an extensive set of novel features and are leveraged by a machine learned approach that depends only on a purely automated training data generation facility. Our methodology can be applied to any other language that has Wikipedia and Web data available (after modifying the simple capitalization rules in Section 4.1). Our resolution models can be easily and quickly retrained with updated data when Wikipedia and the relevant Web data are changed.

For future work, it will be important to investigate other approaches to better predict *oow*. Adding global constraints on resolutions of the same term at multiple locations in the same document may also be important. Of course, developing new features (such as part-of-speech, named entity type, etc) and improving training data quality is always critical, especially for social content sources such as those from Twitter. Finally, directly demonstrating the degree of applicability to other languages is interesting when accounting for the fact that the quality of Wikipedia is variable across languages.

References

Bagga, Amit and Breck Baldwin. 1998. Entity-based cross-document coreferencing using the Vector Space Model. *Proceedings of the 17th international conference on Computational linguistics*.

Bunescu, Razvan and Marius Paşca. 2006. Using Encyclopedic Knowledge for Named Entity Disambiguation. *Proceedings of the 11th Conference of the European Chapter of the Association of Computational Linguistics (EACL-2006)*.

Cucerzan, Silviu. 2007. Large-Scale Named Entity Disambiguation Based on Wikipedia Data. *Pro-*

ceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning.

Fleischman, Ben Michael and Eduard Hovy. 2004. Multi-Document Person Name Resolution. *Proceesing of the Association for Computational Linguistics*.

Friedman, J. H. 2001. Stochastic gradient boosting. *Computational Statistics and Data Analysis,* 38:367–378.

Han, Xianpei and Jun Zhao 2009. Named Entity Disambiguation by Leveraging Wikipedia Semantic Knowledge. *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*.

Mann, S. Gidon and David Yarowsky. 2003. Unsupervised Personal Name Disambiguation. *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*.

Milne, David and Ian H. Witten. 2008a. Learning to Link with Wikipedia. *In Proceedings of the ACM Conference on Information and Knowledge Management (CIKM'2008)*.

Milne, David and Ian H. Witten. 2008b. An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. *Proceedings of the first AAAI Workshop on Wikipedia and Artificial Intelligence*.

Pedersen, Ted, Amruta Purandare and Anagha Kulkarni. 2005. Name Discrimination by Clustering Similar Contexts. *Proceedings of the Sixth International Conference on Intelligent Text Processing and Computational Linguistics (2005)*.

Ravin, Y. and Z. Kazi. 1999. Is Hillary Rodham Clinton the President? In *Association for Computational Linguistics Workshop on Coreference and its Applications*.

Yarowsky, David. 1995. Unsupervised word sense disambiguation rivaling supervised methods. *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics,* pages 189-196.

Zheng, Zhaohui, K. Chen, G. Sun, and H. Zha. 2007. A regression framework for learning ranking functions using relative relevance judgments. *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval,* pages 287-294.

Heterogeneous Parsing via Collaborative Decoding

Muhua Zhu **Jingbo Zhu** **Tong Xiao**
Natural Language Processing Lab.
Northeastern University
zhumuhua@gmail.com
{zhujingbo, xiaotong}@mail.neu.edu.cn

Abstract

There often exist multiple corpora for the same natural language processing (NLP) tasks. However, such corpora are generally used independently due to distinctions in annotation standards. For the purpose of full use of readily available human annotations, it is significant to simultaneously utilize multiple corpora of different annotation standards. In this paper, we focus on the challenge of constituent syntactic parsing with treebanks of different annotations and propose a collaborative decoding (or co-decoding) approach to improve parsing accuracy by leveraging bracket structure consensus between multiple parsing decoders trained on individual treebanks. Experimental results show the effectiveness of the proposed approach, which outperforms state-of-the-art baselines, especially on long sentences.

1 Introduction

Recent years have seen extensive applications of machine learning methods to natural language processing problems. Typically, increase in the scale of training data boosts the performance of machine learning methods, which in turn enhances the quality of learning-based NLP systems (Banko and Brill, 2001). However, annotating data by human is expensive in time and labor. For this reason, human-annotated corpora are considered as the most valuable resource for NLP.

In practice, there often exist more than one corpus for the same NLP tasks. For example, for constituent syntactic parsing (Collins, 1999; Charniak, 2000; Petrov et al., 2006) in Chinese, in addition to the most popular treebank Chinese Treebank (CTB) (Xue et al., 2002), there are also other treebanks such as Tsinghua Chinese Treebank (TCT) (Zhou, 1996). For the purpose of full use of readily available human annotations for the same tasks, it is significant if such corpora can be used jointly. At first sight, a direct combination of multiple corpora is a way to this end. However, corpora created for the same NLP tasks are generally built by different organizations. Thus such corpora often follow different annotation standards and/or even different linguistic theories. We take CTB and TCT as a case study. Although both CTB and TCT are Chomskian-style treebanks, they have annotation divergences in at least two dimensions: a) CTB and TCT have dramatically different tag sets, including parts-of-speech and grammar labels, and the tags cannot be mapped one to one; b) CTB and TCT have distinct hierarchical structures. For example, the words "中国 (Chinese) 传统 (traditional) 文化 (culture)" are grouped as a flat noun phrase according to the CTB standard (right side in Fig. 1), but in TCT, the last two words are instead grouped together beforehand (left side in Fig. 1). The differences cause such treebanks of different annotations to be generally used independently. This paper is dedicated to solving the problem of how to use jointly multiple disparate treebanks for constituent syntactic parsing. Hereafter, treebanks of different annotations are

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1344–1352,
Beijing, August 2010

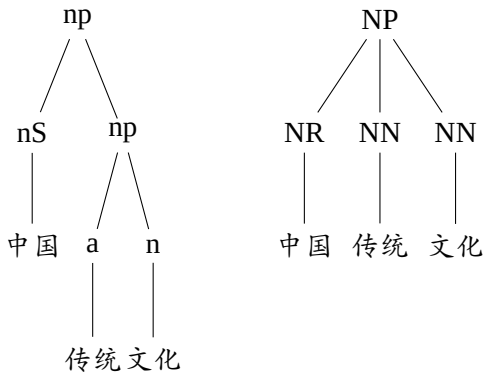
called *heterogeneous treebanks*, and correspondingly, the problem of syntactic parsing with heterogeneous treebanks is referred to as *heterogeneous parsing*.

Previous work on heterogeneous parsing is often based on treebank transformation (or treebank conversion) (Wang et al., 1994; Niu et al., 2009). The basic idea is to transform annotations of one treebank (source treebank) to fit the standard of another treebank (target treebank). Due to divergences of treebank annotations, such transformation is generally achieved in an indirect way by selecting transformation results from the output of a parser trained on the target treebank. A common property of all the work mentioned above is that transformation accuracy is heavily dependent on the performance of parsers trained on the target treebank. Sometimes transformation accuracy is not so satisfactory that techniques like instance pruning are needed in order to refine transformation results (Niu et al., 2009).

We claim there exists another way, interesting but less studied for heterogeneous parsing. The basic idea is that, although there are annotation divergences between heterogenous treebanks, actually we can also find consensus in annotations of bracket structures. Thus we would like to train parsers on individual heterogeneous treebanks and guide the parsers to gain output with consensus in bracket structures as much as possible when they are parsing the same sentences.

To realize this idea, we propose a generic collaborative decoding (or co-decoding) framework where decoders trained on heterogeneous treebanks can exchange consensus information between each other during the decoding phase. Theoretically the framework is able to incorporate a large number of treebanks and various functions that formalize consensus statistics.

Our contributions can be summarized: 1) we propose a co-decoding approach to directly utilizing heterogeneous treebanks; 2) we propose a novel function to measure parsing consensus between multiple decoders. We also conduct experiments on two Chinese treebanks: CTB and TCT. The results show that our approach achieves promising improvements over baseline systems which make no use of consensus information.



中国 传统 文化
(Chinese) (traditional) (culture)

Figure 1: Example tree fragments with TCT (left) and CTB (right) annotations

2 Collaborative Decoding-based Heterogeneous Parsing

2.1 Motivation

This section describes the motivation to use co-decoding for heterogeneous parsing. We first use the example in Fig. 1 to illustrate what consensus information exists between heterogenous treebanks and why such information might help to improve parsing accuracy. This figure contains two partial parse trees corresponding to the words "中国 (Chinese) 传统 (traditional) 文化 (culture)", annotated according to the TCT (left side) and CTB (right side) standards respectively. Despite the distinctions in tag sets and bracket structures, these parse trees actually have partial agreements in bracket structures. That is, not all bracket structures in the parse trees are different. Specifically put, although the internal structures of the parse trees are different, both CTB and TCT agree to take "中国 传统 文化" as a noun phrase. Motivated by this observation, we would like to guide parsers that are trained on CTB and TCT respectively to verify their output interactively by using consensus information implicitly contained in these treebanks. Better performance is expected when such information is considered.

A feasible framework to make use of consensus information is n-best combination (Henderson and Brill, 1999; Sagae and Lavie, 2006; Zhang et al., 2009; Fossum and Knight, 2009). In contrast

to previous work on n-best combination where multiple parsers, say, Collins parser (Collins, 1999) and Berkeley parser (Petrov et al., 2006) are trained on the same training data, n-best combination for heterogeneous parsing is instead allowed to use either a single parser or multiple parsers which are trained on heterogeneous treebanks. Consensus information can be incorporated during the combination of the output (n-best list of full parse trees following distinct annotation standards) of individual parsers. However, despite the success of n-best combination methods, they suffer from the limited scope of n-best list. Taking this into account, we prefer to apply the co-decoding approach such that consensus information is expected to affect the entire procedure of searching hypothesis space.

2.2 System Overview

The idea of co-decoding is recently extensively studied in the literature of SMT (Li et al., 2009; Liu et al., 2009). As the name shows, co-decoding requires multiple decoders be combined and proceed collaboratively. As with n-best combination, there are at least two ways to build multiple decoders: we can either use multiple parsers trained on the same training data (use of diversity of models), or use a single parser on different training data (use of diversity of datasets) [1]. Both ways can build multiple decoders which are to be integrated into co-decoding. For the latter case, one method to get diverse training data is to use different portions of the same training set. In this study we extend the case to an extreme situation where heterogeneous treebanks are used to build multiple decoders.

Fig. 2 represents a basic flow chart of heterogeneous parsing via co-decoding. Note that here we discuss the case of co-decoding with only two decoders, but the framework is generic enough to integrate more than two decoders. For convenience of reference, we call a decoder without incorporating consensus information as *baseline decoder*

[1]To make terminologies clear, we use *parser* as its regular sense, including training models (ex. Collins model 2) and parsing algorithms (ex. the CKY algorithm used in Collins parser), and we use *decoder* to represent parsing algorithms with specified parameter values

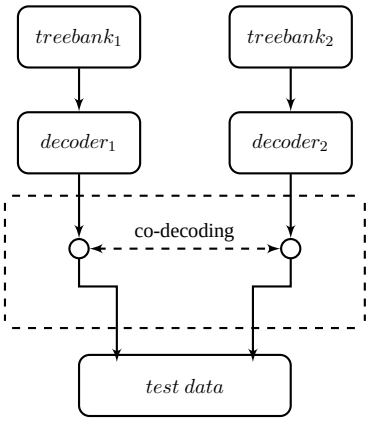


Figure 2: Basic flow chart of co-decoding

and correspondingly refer to a decoder augmented with consensus information as *member decoder*. So the basic steps of co-decoding for heterogeneous parsing is to first build baseline decoders on heterogeneous treebanks and then use the baseline decoders to parse sentences with consensus information exchanged between each other.

To complete co-decoding for heterogeneous parsing, three key components should be considered in the system:

- Co-decoding model. A co-decoder consists of multiple member decoders which are baseline decoders augmented with consensus information. Co-decoding model defines how baseline decoders and consensus information are correlated to get member decoders.

- Decoder coordination. Decoders in the co-decoding model cannot proceed independently but should have interactions between each other in order to exchange consensus information. A decoder coordination strategy decides on when, where, and how the interactions happen.

- Consensus-based score function. Consensus-based score functions formalize consensus information between member decoders. Taking time complexity into consideration, consensus statistics should be able to be computed efficiently.

In the following subsections, we first present the generic co-decoding model and then describe in detail how member decoders collaborate. Finally we introduce a novel consensus-based score function which is used to quantify consensus information exchanged between member decoders.

2.3 Generic Co-decoding Model

The generic co-decoding model described here is also used in (Li et al., 2009) for co-decoding of machine translators. For a given sentence S, a parsing algorithm (decoder) seeks a parse tree T^* which is optimal in the sense that it maximizes some score function $F(T)$, as shown in Eq. 1.

$$T^* = \underset{Ts.t.S=yield(T)}{\arg\max} F(T) \quad (1)$$

where $Ts.t.S = yield(T)$ represents the set of parse trees that yield the input sentence S. For baseline decoders, the score function $F(T)$ is generally just the inside probability $P(T)$ [2] of a tree T, defined as the product of probabilities of grammar rules appearing in parse tree T: $\prod_{r\in R(T)} P(r)$. In the co-decoding framework, $F(T)$ is extended so as to integrate consensus-based score functions which measure consensus information between member decoders, as shown in Eq. 2.

$$F_m(T) = P_m(T) + \sum_{k,k\neq m}^{n} \Psi_k(H_k(S), T) \quad (2)$$

We use d_k to denote the k_{th} decoder and use $H_k(S)$ to denote corresponding parsing hypothesis space of decoder d_k. Moreover, $P_m(T)$ is referred to as *baseline score* given by baseline decoders and $\Psi_k(H_k(S), T)$ is *consensus score* between decoders d_m and d_k, which is defined as a linear combination of consensus-based score functions, as shown in Eq. 3.

$$\Psi_k(H_k(S), T) = \sum_l \lambda_{k,l} f_{k,l}(H_k(S), T) \quad (3)$$

where $f_{k,l}(H_k(S), T)$ represents a consensus-based score function between T and $H_k(S)$, and $\lambda_{k,l}$ is the corresponding weight. Index l ranges over all consensus-based score functions in Eq. 3. Theoretically we can define a variety of consensus-based score functions.

For the simplest case where there are only two member decoders and one consensus-based score function, Eq. 2 and Eq. 3 can be combined and simplified into the equation

$$F_i(T) = P_i(T) + \lambda_{1-i} f(H_{1-i}(S), T) \quad (4)$$

where index i is set to the value of either 1 or 0. This simplified version is used in the experiments of this study.

2.4 Decoder Coordination

This subsection discusses the problem of decoder coordination. Note that although Eq. 2 is defined at sentence level, the co-decoding model actually should be applied to the parsing procedure of any subsequence (word span) of sentence S. So it is natural to render member decoders collaborate when they are processing the same word spans. To this end, we would like to adopt best-first CKY-style parsing algorithms as baseline decoders, since CKY-style decoders have the property that they process word spans in the ascending order of span sizes. Moreover, the hypotheses [3] spanning the same range of words are readily stacked together in a chart cell before CKY-style decoders move on to process other spans. Thus, member decoders can process the same word spans collaboratively from small ones to big ones until they finally complete parsing the entire sentence.

A second issue in Eq. 2 is that consensus-based score functions are dependent on hypothesis space $H_k(S)$. Unfortunately, the whole hypothesis space is not available most of the time. To address this issue, one practical method is to approximate $H_k(S)$ with a n-best hypothesis list. For best-first CKY parsing, we actually retain all unpruned partial hypotheses over the same span as the approximation. Hereafter, the approximation is denoted as $\hat{H}_k(S)$

Finally, we notice in Eq. 2 that consensus score

[2] Actually, the joint probability P(S,T) of sentence S and parse tree T is used, but we can prove that $P(S,T) = P(T)$.

[3] In the literature of syntactic parsing, especially in chart parsing, hypotheses is often called *edges*. This paper will continue to use the terminology *hypothesis* when no ambiguity exists.

$\Psi_k(H_k(S), T)$ and $H_k(S)$ form a circular dependency: searching for $H_k(S)$ requires both baseline score and consensus score; on the other hand, calculating consensus score needs $H_k(S)$ (its approximation in practice) to be known beforehand. Li et al. (2009) solves this dilemma with a bootstrapping method. It starts with seedy n-best lists generated by baseline decoders and then alternates between calculating consensus scores and updating n-best hypothesis lists. Such bootstrapping method is a natural choice to break down the circular dependency, but multi-pass re-decoding might dramatically reduce decoding efficiency. Actually, Li et al. (2009) restricts the iteration number to two in their experiments. In this paper, we instead use an alternative to the bootstrapping method. The process is described as follows.

1. In traditional best-first CKY-style parsing algorithms, hypotheses over the same word spans are grouped according to some criterion of hypothesis equivalence [4]. Among equivalent hypotheses, only a single optimal hypothesis is retained. In this paper, we instead keep top k of equivalent hypotheses in a data structure called *best-first cache*.

2. Use hypotheses in best-first caches to approximate $H_k(S)$, and calculate consensus score $\Psi_k(H_k(S), T)$ between decoders.

3. Use baseline score and consensus score to locally rerank hypotheses in best-first caches. Then remove hypotheses in caches except the top one hypothesis.

In this study, we choose the best-first CKY-style parsing algorithm used in Collins parser (Collins, 1999). Algorithm 1 extends this algorithm for co-decoding. The first two steps initialize baseline decoders and assign appropriate POS tags to sentence S^t. Since baseline decoders are built on heterogeneous treebanks, POS taggers corresponding to each baseline decoder are demanded, unless gold POS tags are provided. The third step is the core of the co-decoding algorithm. Here the *complete* procedure invokes baseline decoders to complete parsing on the span $[start, end]$ and generates $\hat{H}_k(s)$. The *co-decoding* procedure calculates consensus score and locally reranks hypotheses in best-first caches.

[4] the simplest criterion of equivalence is whether hypotheses have the same grammar labels.

Algorithm 1 CKY-style Co-decoding
Argument: d_k {the set of baseline decoders}
S^t {a sentence to be parsed}

Begin
Steps:
1. assign POS tags to sentence S^t
2. initialize baseline decoders d_k
3. for span from 2 to sentence_length **do**
  for start from 1 to (sentence_length-span+1) **do**
    end := (start + span - 1)
    for each base decoder d_k **do**
      complete(d_k, start, end)
    do co-decoding(start, end)
End

Subroutine:
complete(d_k, start, end): base decoder d_k generates hypotheses over the span (begin.end), and fills in best-first caches.
co-decoding(start, end): calculate consensus score and rerank hypotheses in best-first caches. The top 1 is chosen to be the best-first hypothesis.

2.5 Consensus-based Score Function

There are at least two feasible ways to measure consensus between constituency parse trees. By viewing parse trees from diverse perspectives, we can either use functions on bracket structures of parse trees, as in (Wang et al., 1994), or use functions on head-dependent relations by first transforming constituency trees into dependency trees, as in (Niu et al., 2009). Although the co-decoding model is generic enough to integrate various consensus-based score functions in a uniform way, this paper only uses a bracket structure-based function.

As mentioned above, the function proposed in (Wang et al., 1994) is based on bracket structures. Unfortunately, that function is not applicable in the situation of this paper. The reason is that, the function in (Wang et al., 1994) is defined to work on two parse trees, but this paper instead needs a function on a tree T and a set of trees (the approximation $\hat{H}_k(S)$). To this end, we first introduce the concept of *constituent set (CS)* of a parse tree. Conceptually, CS of a parse tree is a set of word spans corresponding to all the sub-

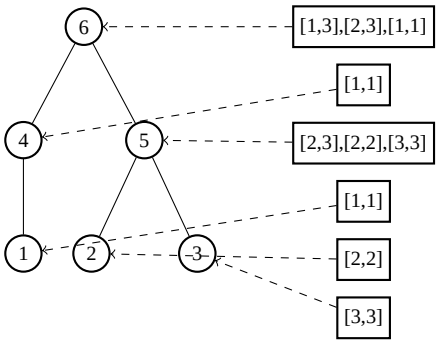


Figure 3: Constituent set of a synthetic parse tree

trees of the tree, as illustrated in Fig. 3. For example, the constituent set of the tree rooted at node 6 has three elements: $[1,1]$, $[1,3]$, and $[1,2]$. For $\hat{H}_k(S)$, the constituent set is defined as the union of constituent sets of all elements it contains.

$$CS(\hat{H}_k(S)) = \bigcup_{T \in \hat{H}_k(S)} CS(T)$$

In practice, we need to cut off elements in $CS(\hat{H}_k(S))$ in order to retain most confident word spans.

With the concept of constituent set, a consensus-based score function on T and $\hat{H}_k(S)$ can be defined as follows.

$$f(\hat{H}_k(S), T) = \frac{\sum_{c \in CS(T)} I(c, CS(\hat{H}_k(S)))}{|CS(T)|} \quad (5)$$

where $I(c, CS(\hat{H}_k(S)))$ is an indicator function which returns one if $c \in CS(T)$ is compatible with all the elements in $CS(\hat{H}_k(S))$, zero otherwise. Two spans, $[a,b]$ and $[i,j]$ are said to be compatible if they satisfy one of the following conditions: 1) $i > b$; 2) $a > j$; 3) $a \leq i \leq b$ and $j \leq b$; 4) $i \leq a \leq j$ and $b \leq j$. Fig 4 uses two example to illustrate the concept of compatibility.

3 Experiments

3.1 Data and Performance Metric

The most recent version of the CTB corpus, CTB 6.0 and the CIPS ParsEval data are used as heterogeneous treebanks in the experiments. Following the split utilized in (Huang et al., 2007), we divided the dataset into blocks of 10 files. For each

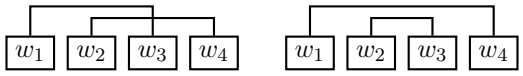


Figure 4: left) two spans conflict; right) two spans are compatible

block, the first file was added to the CTB development data, the second file was added to the CTB testing data, and the remaining 8 files were added to the CTB training data. For the sake of parsing efficiency, we randomly sampled 1,000 sentences of no more than 40 words from the CTB test set.

CTB-Partitions	Train	Dev	Test
#Sentences	22,724	2,855	1,000
#Words	627,833	78,653	25,100
Ave-Length	30.1	30.0	20.3
TCT-Partitions	Train	Dev	Test
#Sentences	32,771	N/A	1,000
#Words	354,767	N/A	10,400
Ave-Length	10.6	N/A	10.4

Table 1: Basic statistics on the CTB and TCT data

CIPS-ParsEval data is publicly available for the first Chinese syntactic parsing competition, CIPS-ParsEval 2009. Compared to CTB, sentences in CIPS-ParsEval data are much shorter in length. We removed sentences which have words less than three. CIPS-ParsEval test set has 7,995 sentences after sentence pruning. As with the CTB test set, we randomly sampled 1,000 sentences for evaluating co-decoding performance. Since CIPS-ParsEval data is actually a portion of the TCT corpus, for convenience of reference, we will refer to CIPS-ParsEval data as TCT in the following sections. Table 1 contains statistics on CTB and TCT.

The two training sets are used individually to build baseline decoders. With regard to the test sets, each sentence in the test sets should have two kinds of POS tags, according to the CTB and TCT standards respectively. To this end, we applied a HMM-based method for POS annotation transformation (Zhu and Zhu, 2009). During the POS transformation, the divergences of word segmentation are omitted.

For all experiments, *bracketing F1* is used as the performance metric, provided by *EVALB* [5].

[5] http://nlp.cs.nyu.edu/evalb

3.2 Baseline Decoders

As already mentioned above, we apply Collins parser in this paper. Specifically speaking, two CKY-style baseline decoders to participate co-decoding are built on CTB and TCT respectively with Collins model two. For the CTB-based decoder, we use the CTB training data with slight modifications: we replaced POS tags of punctuations with specific punctuation symbols.

To get the TCT-based decoder, we made following modifications. Firstly, TCT is available with manually annotated head indices for all the constituents in parse trees. For example, a grammar label, say, np-1, means that the constituent is a noun phrase with the second child being its head child. In order to relax context independence assumptions made in PCFG, we appended head indices to grammar labels to get new labels, for example $np1$. Secondly, since Collins parser is a lexicalized parser, head rules specific to the TCT corpus were manually created, which are used together with readily available head indices. Such adaptation is also used in (Chen et al., 2009);

3.3 Parsing Results

We conduct experiments on both CTB and TCT test sets. Two parameters need to be set: the cut-off threshold for constructing constituent set of $\hat{H}_k(S)$ and the weight λ [6] of consensus score in Eq. 4. We tuned the parameters on the CTB development set and finally set them to 5 and 20 respectively in the experiments. Table 2 presents bracketing F1 scores of baseline systems and the co-decoding approach. Here, the row of *baseline* represents the performance of individual baseline decoders, and the comparison of baseline and co-decoding on a test set, say CTB, demonstrates how much boosting the other side, say TCT, can supply. For the co-decoding approach, the size of best-first cache is set to 5 which achieves the best result among the cache sizes we have experimented.

As the results show, co-decoding achieves promising improvements over baseline systems on both test sets. Interestingly, we see that the improvement on the TCT test set is larger than that on the CTB test set. In general, a relatively strong decoder can improve co-decoding performance more than a relatively weak decoder does. At the first sight, the TCT-based decoder seems to have better performance than the CTB-based decoder. But if taking sentence length into consideration, we can find that the TCT-based decoder is actually relatively weak. Table 3 shows the performance of the CTB-based decoder on short sentences.

[6] We use the same λ for both member decoders.

Test Set	CTB	TCT
Baseline	79.82	81.02
Co-decoding	80.33	81.77

Table 2: Baseline and Co-decoding on the CTB and TCT test sets

3.4 Analysis

Fig. 5 shows the bracketing F1 on the CTB test set at different settings of the best-first cache size C. F1 scores reach the peak before C increases to 6. As a result, we set C to 5 in all our experiments.

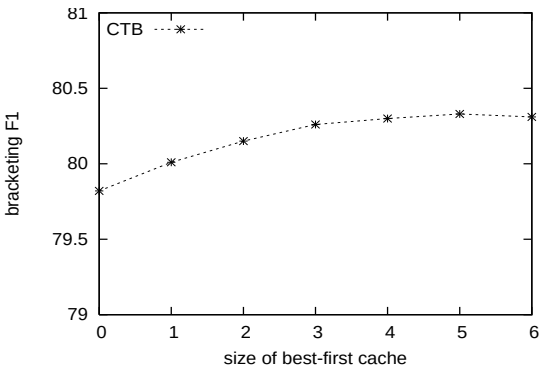


Figure 5: Bracketing F1 with varying best-first cache size

To evaluate the effect of sentence length on co-decoding, Table 3 presents F1 scores on portions of the CTB test set, partitioned according to sentence length. From the results we can see that co-decoding performs better on long sentences. One possible reason is that member decoders have more consensus on big spans. Taking this observation into consideration, one enhancement to the co-decoding approach is to enable co-decoding only on long sentences. This way, parsing ef-

Partitions	[0,10]	(10,20]	(20,30]	(30,40]
# Sentence	276	254	266	204
Ave-Length	6.07	15.64	25.43	35.20
Baseline	92.83	84.34	78.98	76.69
Co-decoding	92.84	84.36	79.43	77.65

Table 3: Effect of sentence length on co-decoding performance

ficiency of co-decoding can be improved. It is worth emphasizing that co-decoding is still helpful for parsers whose performance on short sentences is not satisfactory, as shown in Table 2.

Another interesting analysis is to check how many parsing results are affected by co-decoding, compared to baseline decoders. Table 4 shows the statistics.

Test Set	# All	# Improved	# Decreased
CTB	1000	225	109
TCT	1000	263	92

Table 4: Statistics on sentences of test data

As the table shows, although overall accuracy is increased, we find that on some sentences, co-decoding instead worsens parsing accuracy. In order to get insights on error sources, we manually analyzed 20 sentences on which co-decoding achieves negative results. We find a large portion (14 of 20) of sentences are short sentences (of words less than 20). Actually, due to high accuracy of the CTB-based decoder on short sentences, co-decoding is indifferent when this decoder is processing short sentences. And we also find that some errors are derived from differences in annotation standards. Fortunately, the divergence of annotations mainly exists in relatively small spans. So one solution to the problem is to enable co-decoding on relatively big spans. These will be done in our future work.

4 Related Work

4.1 System Combination

In the literature of syntactic parsing, n-best combination methods include parse selection, constituent recombination, production recombination, and n-best reranking. Henderson and Brill (1999) performs parse selection by maximizing the expected precision of selected parse with respect to the set of parses to be combined. Sagae and Lavie (2006) proposes to recombine constituents from the output of individual parsers. More recently, Fossum and Knight (2009) studies a combination method at production level. Zhang et al. (2009) reranks n-best list of one parser with scores derived from another parser.

Compared to n-best combination, co-decoding (Li et al., 2009; Liu et al., 2009) combines systems during decoding phase. Theoretically, system combination during decoding phase helps decoders to select better approximation to hypothesis space, since pruning is practically unavoidable. To the best of our knowledge, co-decoding methods have not been applied to syntactic parsing.

4.2 Treebank Transformation

The focus of this study is heterogeneous parsing. Previous work on this challenge is generally based on treebank transformation. Wang et al. (1994) describes a method for transformation between constituency treebanks. The basic idea is to train a parser on a target treebank and generate a n-best list for each sentence in source treebank(s). Then, a matching metric which is a function on the number of the same word spans between two trees is defined to select a best parse from each n-best list. Niu et al. (2009) applies a closely similar framework as with (Wang et al., 1994) to transform a dependency treebank to a constituency one.

5 Conclusions

This paper proposed a co-decoding approach to the challenge of heterogeneous parsing. Compared to previous work on this challenge, co-decoding is able to directly utilize heterogeneous treebanks by incorporating consensus information between partial output of individual parsers during the decoding phase. Experiments demonstrate the effectiveness of the co-decoding approach, especially the effectiveness on long sentences.

Acknowledgments

This work was supported in part by the National Science Foundation of China (60873091). We would like to thank our anonymous reviewers for their comments.

References

Banko, Michele and Eric Brill. 2001. *Scaling to very very large corpora for natural language disambiguation*. In Proc. of ACL 2001, pages 26-33.

Chen, Xiao, Changning Huang, Mu Li, and Chunyu Kit. 2009. *Better Parser Combination*. Technique Report of CIPS-ParsEval 2009.

Collins, Michael. 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis, University of Pennsylvania.

Charniak, Eugene. 2000. *A maximum-entropy-inspired parser*. In Proc. of NAACL 2000, pages 132-139.

Fossum, Victoria and Kevin Knight. 2009. *Combining constituent parsers*. In Proc. of NAACL 2009, pages 253-256.

Henderson, John and Eric Brill. 1999. *Exploiting diversity in natural language processing*. In Proc. of SIGDAT-EMNLP 1999, pages 187-194.

Huang, Zhongqiang, Mary P. Harper, and Wen Wang. 2007. *Mandarin part-of-speech tagging and discriminative reranking*. In Proc. of EMNLP-CoNLL 2007, pages 1093-1102.

Li, Mu, Nan Duan, Dongdong Zhang, Chi-Ho Li, and Ming Zhou. 2009. *Collaborative decoding: partial hypothesis re-ranking using trnaslationconsensus between decoders*. In Proc. of ACL 2009, pages 585-592.

Liu, Yang, Haitao Mi, Yang Feng, and Qun Liu. 2009. *Joint Decoding with Multiple Translation Models*. In Proc. of ACL 2009, pages 576-584.

Niu, Zheng-Yu, Haifeng Wang, Hua Wu. 2009. *Exploiting heterogeneous treebanks for parsing*. In Proc. of ACL 2009, pages 46-54.

Petrov, Slav, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. *Learning accurate, compact, and interpretable tree annotation*. In Proc. of COLING-ACL 2006, pages 433-440.

Sage, Kenji and Alon Lavie. 2006. *Parser combination by reparsing*. In Proc. of NAACL 2006, pages 129-132.

Xue, Nianwen, Fu dong Chiou, and Martha Palmer. 2002. *Building a large-scale Annotated Chinese corpus*. In Proc. of COLING 2002, pages 1-8.

Wang, Jong-Nae, Jing-Shin Chang, and Keh-Yih Su. 1994. *An automatic treebank conversion algorithm for corpus sharing*. In Proc. of ACL 1994, pages 248-254.

Zhang, Hui, Min Zhang, Chew Lim Tan, and Haizhou Li. 2009. *K-best combination of syntactic parsers*. In Proc. of EMNLP 2009, pages 1552-1560.

Zhou, Qiang. 1996. *Phrase bracketing and annotating on Chinese language corpus. (in Chinese)* Ph.D. thesis, Beijing University.

Zhu, Muhua and Jingbo Zhu. 2009. *Label Correspondence Learning for Part-of-Speech Annotation Transformation*. In Proc. of CIKM 2009, pages 1461-1464.

A Monolingual Tree-based Translation Model for Sentence Simplification*

Zhemin Zhu[1]
Department of Computer Science
Technische Universität Darmstadt
[1]http://www.ukp.tu-darmstadt.de

Delphine Bernhard[2]
LIMSI-CNRS
[2]delphine.bernhard@limsi.fr

Iryna Gurevych[1]
Department of Computer Science
Technische Universität Darmstadt

Abstract

In this paper, we consider sentence simplification as a special form of translation with the complex sentence as the source and the simple sentence as the target. We propose a Tree-based Simplification Model (TSM), which, to our knowledge, is the first statistical simplification model covering splitting, dropping, reordering and substitution integrally. We also describe an efficient method to train our model with a large-scale parallel dataset obtained from the Wikipedia and Simple Wikipedia. The evaluation shows that our model achieves better readability scores than a set of baseline systems.

1 Introduction

Sentence simplification transforms long and difficult sentences into shorter and more readable ones. This helps humans read texts more easily and faster. Reading assistance is thus an important application of sentence simplification, especially for people with reading disabilities (Carroll et al., 1999; Inui et al., 2003), low-literacy readers (Watanabe et al., 2009), or non-native speakers (Siddharthan, 2002).

Not only human readers but also NLP applications can benefit from sentence simplification. The original motivation for sentence simplification is using it as a preprocessor to facilitate parsing or translation tasks (Chandrasekar et al., 1996). Complex sentences are considered as stumbling blocks for such systems. More recently, sentence simplification has also been shown helpful for summarization (Knight and Marcu, 2000), sentence fusion (Filippova and Strube, 2008b), semantic role labeling (Vickrey and Koller, 2008), question generation (Heilman and Smith, 2009), paraphrase generation (Zhao et al., 2009) and biomedical information extraction (Jonnalagadda and Gonzalez, 2009).

At sentence level, reading difficulty stems either from lexical or syntactic complexity. Sentence simplification can therefore be classified into two types: *lexical simplification* and *syntactic simplification* (Carroll et al., 1999). These two types of simplification can be further implemented by a set of simplification operations. *Splitting*, *dropping*, *reordering*, and *substitution* are widely accepted as important simplification operations. The splitting operation splits a long sentence into several shorter sentences to decrease the complexity of the long sentence. The dropping operation further removes unimportant parts of a sentence to make it more concise. The reordering operation interchanges the order of the split sentences (Siddharthan, 2006) or parts in a sentence (Watanabe et al., 2009). Finally, the substitution operation replaces difficult phrases or words with their simpler synonyms.

In most cases, different simplification operations happen simultaneously. It is therefore necessary to consider the simplification process as a combination of different operations and treat them as a whole. However, most of the existing models only consider one of these operations. Siddharthan (2006) and Petersen and Ostendorf (2007) focus on sentence splitting, while sentence compression systems (Filippova and Strube, 2008a) mainly use the dropping operation. As far as lexical simplification is concerned, word substitution is usually done by selecting simpler synonyms from Wordnet based on word frequency (Carroll et al., 1999).

In this paper, we propose a sentence simplification model by tree transformation which is based

* This work has been supported by the Emmy Noether Program of the German Research Foundation (DFG) under the grant No. GU 798/3-1, and by the Volkswagen Foundation as part of the Lichtenberg-Professorship Program under the grant No. I/82806.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1353–1361,
Beijing, August 2010

on techniques from statistical machine translation (SMT) (Yamada and Knight, 2001; Yamada and Knight, 2002; Graehl et al., 2008). Our model integrally covers splitting, dropping, reordering and phrase/word substitution. The parameters of our model can be efficiently learned from complex-simple parallel datasets. The transformation from a complex sentence to a simple sentence is conducted by applying a sequence of simplification operations. An expectation maximization (EM) algorithm is used to iteratively train our model. We also propose a method based on monolingual word mapping which speeds up the training process significantly. Finally, a decoder is designed to generate the simplified sentences using a greedy strategy and integrates language models.

In order to train our model, we further compile a large-scale complex-simple parallel dataset (PWKP) from Simple English Wikipedia[1] and English Wikipedia[2], as such datasets are rare.

We organize the remainder of the paper as follows: Section 2 describes the PWKP dataset. Section 3 presents our TSM model. Sections 4 and 5 are devoted to training and decoding, respectively. Section 6 details the evaluation. The conclusions follow in the final section.

2 Wikipedia Dataset: PWKP

We collected a paired dataset from the English Wikipedia and Simple English Wikipedia. The targeted audience of Simple Wikipedia includes "children and adults who are learning English language". The authors are requested to "use easy words and short sentences" to compose articles. We processed the dataset as follows:

Article Pairing 65,133 articles from Simple Wikipedia[3] and Wikipedia[4] were paired by following the "language link" using the dump files in Wikimedia.[5] Administration articles were further removed.

Plain Text Extraction We use JWPL (Zesch et al., 2008) to extract plain texts from Wikipedia articles by removing specific Wiki tags.

Pre-processing including sentence boundary detection and tokenization with the Stanford Parser package (Klein and Manning, 2003), and lemmatization with the TreeTagger (Schmid, 1994).

Monolingual Sentence Alignment As we need a parallel dataset aligned at the sentence level, we further applied monolingual sentence alignment on the article pairs. In order to achieve the best sentence alignment on our dataset, we tested three similarity measures: (i) sentence-level TF*IDF (Nelken and Shieber, 2006), (ii) word overlap (Barzilay and Elhadad, 2003) and (iii) word-based maximum edit distance (MED) (Levenshtein, 1966) with costs of insertion, deletion and substitution set to 1. To evaluate their performance we manually annotated 120 sentence pairs from the article pairs. Tab. 1 reports the precision and recall of these three measures. We manually adjusted the similarity threshold to obtain a recall value as close as possible to 55.8% which was previously adopted by Nelken and Shieber (2006).

Similarity	Precision	Recall
TF*IDF	91.3%	55.4%
Word Overlap	50.5%	55.1%
MED	13.9%	54.7%

Table 1: Monolingual Sentence Alignment

The results in Tab. 1 show that sentence-level TF*IDF clearly outperforms the other two measures, which is consistent with the results reported by Nelken and Shieber (2006). We henceforth chose sentence-level TF*IDF to align our dataset.

As shown in Tab. 2, PWKP contains more than 108k sentence pairs. The sentences from Wikipedia and Simple Wikipedia are considered as "complex" and "simple" respectively. Both the average sentence length and average token length in Simple Wikipedia are shorter than those in Wikipedia, which is in compliance with the purpose of Simple Wikipedia.

Avg. Sen. Len		Avg. Tok. Len		#Sen.Pairs
complex	simple	complex	simple	-
25.01	20.87	5.06	4.89	108,016

Table 2: Statistics for the PWKP dataset

In order to account for sentence splitting, we allow 1 to n sentence alignment to map one complex sentence to several simple sentences. We first perform 1 to 1 mapping with sentence-level TF*IDF and then combine the pairs with the same complex sentence and adjacent simple sentences.

3 The Simplification Model: TSM

We apply the following simplification operations to the parse tree of a complex sentence: splitting,

[1] http://simple.wikipedia.org
[2] http://en.wikipedia.org
[3] As of Aug 17th, 2009
[4] As of Aug 22nd, 2009
[5] http://download.wikimedia.org

dropping, reordering and substitution. In this section, we use a running example to illustrate this process. c is the complex sentence to be simplified in our example. Fig. 1 shows the parse tree of c (we skip the POS level).

c: *August was the sixth month in the ancient Roman calendar which started in 735BC.*

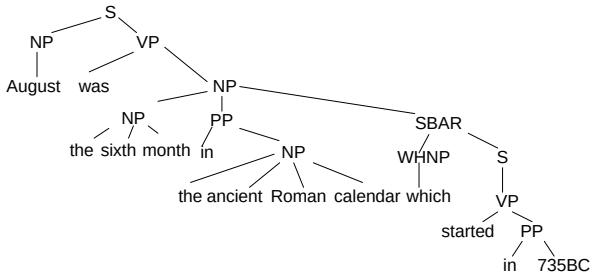


Figure 1: Parse Tree of c

3.1 Splitting

The first operation is sentence splitting, which we further decompose into two subtasks: (i) *segmentation*, which decides where and whether to split a sentence and (ii) *completion*, which makes the new split sentences complete.

First, we decide where we can split a sentence. In our model, the splitting point is judged by the syntactic constituent of the split boundary word in the complex sentence. The decision whether a sentence should be split is based on the length of the complex sentence. The features used in the *segmentation* step are shown in Tab. 3.

Word	Constituent	iLength	isSplit	Prob.
"which"	SBAR	1	true	0.0016
"which"	SBAR	1	false	0.9984
"which"	SBAR	2	true	0.0835
"which"	SBAR	2	false	0.9165

Table 3: Segmentation Feature Table (SFT)

Actually, we do not use the direct constituent of a word in the parse tree. In our example, the direct constituent of the word "which" is "WHNP". Instead, we use Alg. 1 to calculate the constituent of a word. Alg. 1 returns "SBAR" as the adjusted constituent for "which". Moreover, directly using the length of the complex sentence is affected by the data sparseness problem. Instead, we use $\mathrm{iLength}$ as the feature which is calculated as $\mathrm{iLength} = \mathrm{ceiling}(\frac{\mathrm{comLength}}{\mathrm{avgSimLength}})$, where $\mathrm{comLength}$ is the length of the complex sentence and $\mathrm{avgSimLength}$ is the average length of simple sentences in the training dataset. The "Prob." column shows the probabilities obtained after training on our dataset.

Algorithm 1 adjustConstituent($word$, $tree$)

```
constituent ← word.father;
father ← constituent.father;
while father ≠ NULL AND constituent is the most
left child of father do
    constituent ← father;
    father ← father.father;
end while
return constituent;
```

In our model, one complex sentence can be split into two or more sentences. Since many splitting operations are possible, we need to select the most likely one. The probability of a segmentation operation is calculated as:

$$P(seg|c) = \prod_{w:c} SFT(w|c) \quad (1)$$

where w is a word in the complex sentence c and $SFT(w|c)$ is the probability of the word w in the Segmentation Feature Table (SFT); Fig. 2 shows a possible segmentation result of our example.

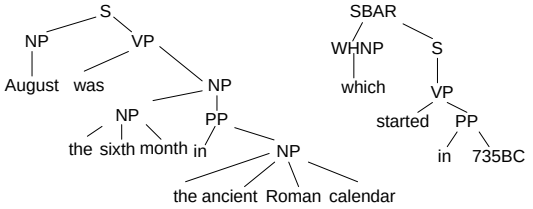


Figure 2: Segmentation

The second step is *completion*. In this step, we try to make the split sentences complete and grammatical. In our example, to make the second sentence "which started in 735BC" complete and grammatical we should first drop the border word "which" and then copy the dependent NP "the ancient Roman calendar" to the left of "started" to obtain the complete sentence "the ancient Roman calendar started in 735BC". In our model, whether the border word should be dropped or retained depends on two features of the border word: the direct constituent of the word and the word itself, as shown in Tab. 4.

Const.	Word	isDropped	Prob.
WHNP	which	True	1.0
WHNP	which	False	Prob.Min

Table 4: Border Drop Feature Table (BDFT)

In order to copy the necessary parts to complete the new sentences, we must decide which parts should be copied and where to put these parts in the new sentences. In our model, this is judged by two features: the dependency relation and the constituent. We use the Stanford Parser for parsing the dependencies. In our example, the de-

pendency relation between "calendar" in the complex sentence and the verb "started" in the second split sentence is "gov_nsubj".[6] The direct constituent of "started" is "VP" and the word "calendar" should be put on the "left" of "started", see Tab. 5.

Dep.	Const.	isCopied	Pos.	Prob.
gov_nsubj	VP(VBD)	True	left	0.9000
gov_nsubj	VP(VBD)	True	right	0.0994
gov_nsubj	VP(VBD)	False	-	0.0006

Table 5: Copy Feature Table (CFT)

For dependent NPs, we copy the whole NP phrase rather than only the head noun.[7] In our example, we copy the whole NP phrase "the ancient Roman calendar" to the new position rather than only the word "calendar". The probability of a completion operation can be calculated as

$$P(com|seg) = \prod_{bw:s} BDFT(bw|s) \prod_{w:s} \prod_{dep:w} CFT(dep).$$

where s are the split sentences, bw is a border word in s, w is a word in s, dep is a dependency of w which is out of the scope of s. Fig. 3 shows the most likely result of the completion operation for our example.

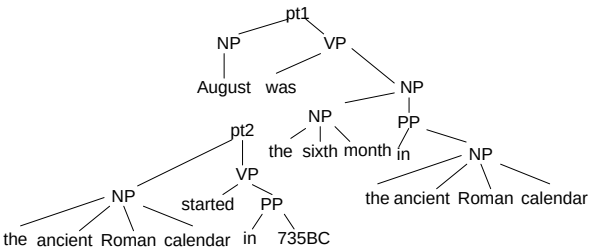


Figure 3: Completion

3.2 Dropping and Reordering

We first apply dropping and then reordering to each non-terminal node in the parse tree from top to bottom. We use the same features for both dropping and reordering: the node's direct constituent and its children's constituents pattern, see Tab. 6 and Tab. 7.

Constituent	Children	Drop	Prob.
NP	DT_JJ_NNP_NN	1101	7.66E-4
NP	DT_JJ_NNP_NN	0001	1.26E-7

Table 6: Dropping Feature Table (DFT)

[6]With Stanford Parser, "which" is a referent of "calender" and the nsubj of "started". "calender" thus can be considered to be the nsubj of "started" with "started" as the governor.

[7]The copied NP phrase can be further simplified in the following steps.

Constituent	Children	Reorder	Prob.
NP	DT_JJ_NN	012	0.8303
NP	DT_JJ_NN	210	0.0039

Table 7: Reordering Feature Table (RFT)

The bits '1' and '0' in the "Drop" column indicate whether the corresponding constituent is retained or dropped. The number in the "Reorder" column represents the new order for the children. The probabilities of the dropping and reordering operations can be calculated as Equ. 2 and Equ. 3.

$$P(dp|node) = DFT(node) \quad (2)$$

$$P(ro|node) = RFT(node) \quad (3)$$

In our example, one of the possible results is dropping the NNP "Roman", as shown in Fig. 4.

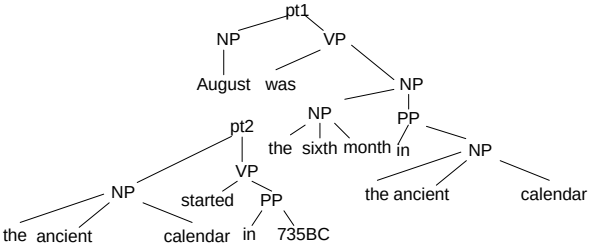


Figure 4: Dropping & Reordering

3.3 Substitution

3.3.1 Word Substitution

Word substitution only happens on the terminal nodes of the parse tree. In our model, the conditioning features include the original word and the substitution. The substitution for a word can be another word or a multi-word expression (see Tab. 8). The probability of a word substitution operation can be calculated as $P(sub|w) = SubFT(Substitution|Origin)$.

Origin	Substitution	Prob.
ancient	ancient	0.963
ancient	old	0.0183
ancient	than transport	1.83E-102
old	ancient	0.005

Table 8: Substitution Feature Table (SubFT)

3.3.2 Phrase Substitution

Phrase substitution happens on the non-terminal nodes and uses the same conditioning features as word substitution. The "Origin" consists of the leaves of the subtree rooted at the node. When we apply phrase substitution on a non-terminal node, then any simplification operation (including dropping, reordering and substitution) cannot happen on its descendants any more

because when a node has been replaced then its descendants are no longer existing. Therefore, for each non-terminal node we must decide whether a substitution should take place at this node or at its descendants. We perform substitution for a non-terminal $node$ if the following constraint is met:

$$Max(SubFT(*|node)) \geq \prod_{ch:node} Max(SubFT(*|ch)).$$

where ch is a child of the $node$. "$*$" can be any substitution in the SubFT. The probability of the phrase substitution is calculated as $P(sub|node) = SubFT(Substitution|Origin)$. Fig. 5 shows one of the possible substitution results for our example where "ancient" is replaced by "old".

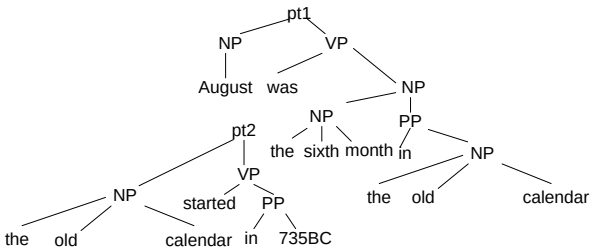


Figure 5: Substitution

As a result of all the simplification operations, we obtain the following two sentences: $s1 = Str(pt1)$="*August was the sixth month in the old calendar.*" and $s2 = Str(pt2)$="*The old calendar started in 735BC.*"

3.4 The Probabilistic Model

Our model can be formalized as a direct translation model from complex to simple $P(s|c)$ multiplied by a language model $P(s)$ as shown in Equ. 4.

$$s = \underset{s}{\operatorname{argmax}} P(s|c)P(s) \quad (4)$$

We combine the parts described in the previous sections to get the direct translation model:

$$P(s|c) = \sum_{\theta:Str(\theta(c))=s} (P(seg|c)P(com|seg) \quad (5)$$
$$\prod_{node} P(dp|node)P(ro|node)P(sub|node)$$
$$\prod_{w} (sub|w)).$$

where θ is a sequence of simplification operations and $Str(\theta(c))$ corresponds to the leaves of a simplified tree. There can be many sequences of operations that result in the same simplified sentence and we sum up all of their probabilities.

4 Training

In this section, we describe how we train the probabilities in the tables. Following the work of Yamada and Knight (2001), we train our model by maximizing $P(s|c)$ over the training corpus with the EM algorithm described in Alg. 2, using a constructed graph structure. We develop the Training Tree (Fig. 6) to calculate $P(s|c)$. $P(s|c)$ is equal to the inside probability of the root in the Training Tree. Alg. 3 and Alg. 4 are used to calculate the inside and outside probabilities. We refer readers to Yamada and Knight (2001) for more details.

Algorithm 2 EM Training ($dataset$)

```
Initialize all probability tables using the uniform distribution;
for several iterations do
    reset all cnt = 0;
    for each sentence pair < c, s > in dataset do
        tt = buildTrainingTree(< c, s >);
        calcInsideProb(tt);
        calcOutsideProb(tt);
        update cnt for each conditioning feature in each
        node of tt: cnt = cnt + node.insideProb *
        node.outsideProb/root.insideProb;
    end for
    updateProbability();
end for
```

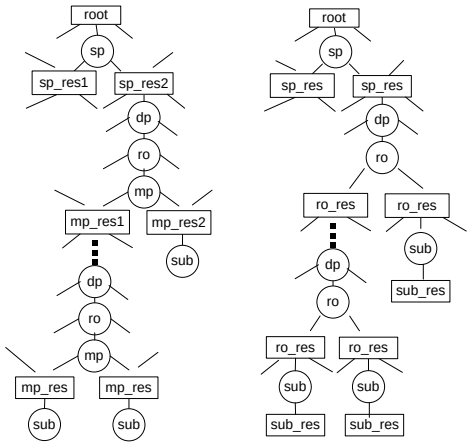


Figure 6: Training Tree (Left) and Decoding Tree (Right)

We illustrate the construction of the training tree with our running example. There are two kinds of nodes in the training tree: data nodes in rectangles and operation nodes in circles. Data nodes contain data and operation nodes execute operations. The training is a supervised learning

process with the parse tree of c as input and the two strings $s1$ and $s2$ as the desired output. $root$ stores the parse tree of c and also $s1$ and $s2$. sp, ro, mp and sub are splitting, reordering, mapping and substitution operations. sp_res and mp_res store the results of sp and mp. In our example, sp splits the parse tree into two parse trees $pt1$ and $pt2$ (Fig. 3). sp_res1 contains $pt1$ and $s1$. sp_res2 contains $pt2$ and $s2$. Then dp, ro and mp are iteratively applied to each non-terminal node at each level of $pt1$ and $pt2$ from top to down. This process continues until the terminal nodes are reached or is stopped by a sub node. The function of mp operation is similar to the word mapping operation in the string-based machine translation. It maps substrings in the complex sentence which are dominated by the children of the current node to proper substrings in the simple sentences.

Speeding Up The example above is only one of the possible paths. We try all of the promising paths in training. Promising paths are the paths which are likely to succeed in transforming the parse tree of c into $s1$ and $s2$. We select the promising candidates using monolingual word mapping as shown in Fig. 7. In this example, only the word "which" can be a promising candidate for splitting. We can select the promising candidates for the dropping, reordering and mapping operations similarly. With this improvement, we can train on the PWKP dataset within 1 hour excluding the parsing time taken by the Stanford Parser.

We initialize the probabilities with the uniform distribution. The binary features, such as SFT and BDFT, are assigned the initial value of 0.5. For DFT and RFT, the initial probability is $\frac{1}{N!}$, where N is the number of the children. CFT is initialized as 0.25. SubFT is initialized as 1.0 for any substitution at the first iteration. After each iteration, the $updateProbability$ function recalculates these probabilities based on the cnt for each feature.

Algorithm 3 calcInsideProb (TrainingTree tt)

for each $node$ from $level = N$ to $root$ of tt **do**
  if $node$ is a sub node **then**
    $node.insideProb$ = $P(sub|node)$;
  else if $node$ is a mp OR sp node **then**
    $node.insideProb$ = $\prod_{child} child.insideProb$;
  else
    $node.insideProb$ = $\sum_{child} child.insideProb$;
  end if
end for

Algorithm 4 calcOutsideProb (TrainingTree tt)

for each $node$ from $root$ to $level = N$ of tt **do**
  if $node$ is the $root$ **then**
    $node.outsideProb$ = 1.0;
  else if $node$ is a sp_res OR mp_res node **then**
    {COMMENT: $father$ are the fathers of the current $node$, $sibling$ are the children of $father$ excluding the current $node$}
    $node.outsideProb = \sum_{father} father.outsideProb * \prod_{sibling} sibling.insideProb$;
  else if $node$ is a mp node **then**
    $node.outsideProb$ = $father.outsideProb * 1.0$;
  else if $node$ is a sp, ro, dp or sub node **then**
    $node.outsideProb = father.outsideProb * P(sp$ or ro or dp or $sub|node)$;
  end if
end for

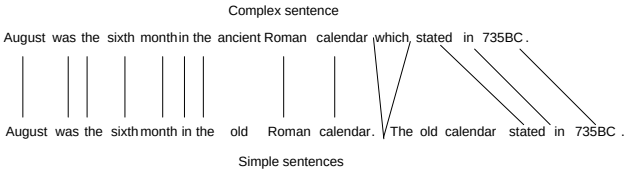


Figure 7: Monolingual Word Mapping

5 Decoding

For decoding, we construct the decoding tree (Fig. 6) similarly to the construction of the training tree. The decoding tree does not have mp operations and there can be more than one sub nodes attached to a single ro_res. The $root$ contains the parse tree of the complex sentence. Due to space limitations, we cannot provide all the details of the decoder.

We calculate the inside probability and outside probability for each node in the decoding tree. When we simplify a complex sentence, we start from the root and greedily select the branch with the highest outside probability. For the substitution operation, we also integrate a trigram language model to make the generated sentences more fluent. We train the language model with SRILM (Stolcke, 2002). All the articles from the Simple Wikipedia are used as the training corpus, amounting to about 54 MB.

6 Evaluation

Our evaluation dataset consists of 100 complex sentences and 131 parallel simple sentences from PWKP. They have not been used for training. Four baseline systems are compared in our evaluation. The first is Moses which is a state of the art SMT system widely used as a baseline in MT community. Obviously, the purpose of Moses is cross-lingual translation rather than monolin-

gual simplification. The goal of our comparison is therefore to assess how well a standard SMT system may perform simplification when fed with a proper training dataset. We train Moses with the same part of PWKP as our model. The second baseline system is a sentence compression system (Filippova and Strube, 2008a) whose demo system is available online.[8] As the compression system can only perform dropping, we further extend it to our third and fourth baseline systems, in order to make a reasonable comparison. In our third baseline system, we substitute the words in the output of the compression system with their simpler synonyms. This is done by looking up the synonyms in Wordnet and selecting the most frequent synonym for replacement. The word frequency is counted using the articles from Simple Wikipedia. The fourth system performs sentence splitting on the output of the third system. This is simply done by splitting the sentences at "and", "or", "but", "which", "who" and "that", and discarding the border words. In total, there are 5 systems in our evaluation: *Moses*, the MT system; *C*, the compression system; *CS*, the compression+substitution system; *CSS*, the compression+substitution+split system; *TSM*, our model. We also provide evaluation measures for the sentences in the evaluation dataset: *CW*: complex sentences from Normal Wikipedia and *SW*: parallel simple sentences from Simple Wikipedia.

6.1 Basic Statistics and Examples

The first three columns in Tab. 9 present the basic statistics for the evaluation sentences and the output of the five systems. $tokenLen$ is the average length of tokens which may roughly reflect the lexical difficulty. TSM achieves an average token length which is the same as the Simple Wikipedia (SW). $senLen$ is the average number of tokens in one sentence, which may roughly reflect the syntactic complexity. Both TSM and CSS produce shorter sentences than SW. Moses is very close to CW. $\#sen$ gives the number of sentences. Moses, C and CS cannot split sentences and thus produce about the same number of sentences as available in CW.

Here are two example results obtained with our TSM system.

Example 1. *CW*: "Genetic engineering has expanded the genes available to breeders to utilize in creating desired germlines for new crops." *SW*: "New plants were created with genetic engineering." *TSM*: "Engineering has expanded the genes available to breeders to use in making germlines for new crops."

Example 2. *CW*: "An umbrella term is a word that provides a superset or grouping of related concepts, also called a hypernym." *SW*: "An umbrella term is a word that provides a superset or grouping of related concepts." *TSM*: "An umbrella term is a word. A word provides a superset of related concepts, called a hypernym."

In the first example, both substitution and dropping happen. TSM replaces "utilize" and "creating" with "use" and "making". "Genetic" is dropped. In the second example, the complex sentence is split and "also" is dropped.

6.2 Translation Assessment

In this part of the evaluation, we use traditional measures used for evaluating MT systems. Tab. 9 shows the BLEU and NIST scores. We use "mteval-v11b.pl"[9] as the evaluation tool. CW and SW are used respectively as source and reference sentences. TSM obtains a very high BLEU score (0.38) but not as high as Moses (0.55). However, the original complex sentences (CW) from Normal Wikipedia get a rather high BLEU (0.50), when compared to the simple sentences. We also find that most of the sentences generated by Moses are exactly the same as those in CW: this shows that Moses only performs few modifications to the original complex sentences. This is confirmed by MT evaluation measures: if we set CW as both source and reference, the BLEU score obtained by Moses is 0.78. TSM gets 0.55 in the same setting which is significantly smaller than Moses and demonstrates that TSM is able to generate simplifications with a greater amount of variation from the original sentence. As shown in the "#Same" column of Tab. 9, 25 sentences generated by Moses are exactly identical to the complex sentences, while the number for TSM is 2 which is closer to SW. It is however not clear how well BLEU and NIST discriminate simplification systems. As discussed in Jurafsky and Martin (2008), "BLEU does poorly at comparing systems with radically different architectures and is most appropriate when evaluating incremental changes with similar architectures." In our case, TSM and CSS can be considered as having similar architectures as both of them can do splitting, dropping

[8] http://212.126.215.106/compression/

[9] http://www.statmt.org/moses/

	TokLen	SenLen	#Sen	BLEU	NIST	#Same	Flesch	Lix(Grade)	OOV%	PPL
CW	4.95	27.81	100	0.50	6.89	100	49.1	53.0 (10)	52.9	384
SW	4.76	17.86	131	1.00	10.98	3	60.4 (PE)	44.1 (8)	50.7	179
Moses	4.81	26.08	100	**0.55**	**7.47**	25	54.8	48.1 (9)	52.0	363
C	4.98	18.02	103	0.28	5.37	1	56.2	45.9 (8)	51.7	481
CS	4.90	18.11	103	0.19	4.51	0	59.1	45.1 (8)	**49.5**	616
CSS	4.98	**10.20**	**182**	0.18	4.42	0	65.5 (PE)	38.3 (6)	53.4	581
TSM	**4.76**	13.57	180	0.38	6.21	**2**	**67.4 (PE)**	**36.7 (5)**	50.8	**353**

Table 9: Evaluation

and substitution. But Moses mostly cannot split and drop. We may conclude that TSM and Moses have different architectures and BLEU or NIST is not suitable for comparing them. Here is an example to illustrate this: *(CW):* "Almost as soon as he leaves, Annius and the guard Publius arrive to **escort** Vitellia to Titus, who has now chosen her as his empress." *(SW):* "Almost as soon as he leaves, Annius and the guard Publius arrive to **take** Vitellia to Titus, who has now chosen her as his empress." *(Moses):* The same as (SW). *(TSM):* "Annius and the guard Publius arrive to **take** Vitellia to Titus. Titus has now chosen her as his empress." In this example, *Moses* generates an exactly identical sentence to *SW*, thus the BLUE and NIST scores of *Moses* is the highest. *TSM* simplifies the complex sentence by dropping, splitting and substitution, which results in two sentences that are quite different from the *SW* sentence and thus gets lower BLUE and NIST scores. Nevertheless, the sentences generated by *TSM* seem better than *Moses* in terms of simplification.

6.3 Readability Assessment

Intuitively, readability scores should be suitable metrics for simplification systems. We use the Linux "style" command to calculate the Flesch and Lix readability scores. The results are presented in Tab. 9. "PE" in the Flesch column stands for "Plain English" and the "Grade" in Lix represents the school year. TSM achieves significantly better scores than Moses which has the best BLEU score. This implies that good monolingual translation is not necessarily good simplification. OOV is the percentage of words that are not in the Basic English BE850 list.[10] TSM is ranked as the second best system for this criterion.

The perplexity (PPL) is a score of text probability measured by a language model and normalized by the number of words in the text (Equ. 6). PPL can be used to measure how tight the language model fits the text. Language models constitute an important feature for assessing readability (Schwarm and Ostendorf, 2005). We train a trigram LM using the simple sentences in PWKP and calculate the PPL with SRILM. TSM gets the best PPL score. From this table, we can conclude that TSM achieves better overall readability than the baseline systems.

$$PPL(text) = P(w_1 w_2 ... w_N)^{-\frac{1}{N}} \quad (6)$$

There are still some important issues to be considered in future. Based on our observations, the current model performs well for word substitution and segmentation. But the completion of the new sentences is still problematic. For example, we copy the dependent NP to the new sentences. This may break the coherence between sentences. A better solution would be to use a pronoun to replace the NP. Sometimes, excessive droppings occur, e.g., "older" and "twin" are dropped in "She has an older brother and a twin brother...". This results in a problematic sentence: "She has an brother and a brother...". There are also some errors which stem from the dependency parser. In Example 2, "An umbrella term" should be a dependency of "called". But the parser returns "superset" as the dependency. In the future, we will investigate more sophisticated features and rules to enhance TSM.

7 Conclusions

In this paper, we presented a novel large-scale parallel dataset PWKP for sentence simplification. We proposed TSM, a tree-based translation model for sentence simplification which covers splitting, dropping, reordering and word/phrase substitution integrally for the first time. We also described an efficient training method with speeding up techniques for TSM. The evaluation shows that TSM can achieve better overall readability scores than a set of baseline systems.

[10] http://simple.wikipedia.org/wiki/Wikipedia:Basic_English_alphabetical_wordlist

References

Barzilay, Regina and Noemie Elhadad. 2003. Sentence alignment for monolingual comparable corpora. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 25–32.

Carroll, John, Guido Minnen, Darren Pearce, Yvonne Canning, Siobhan Devlin, and John Tait. 1999. Simplifying text for language-impaired readers. In *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics (EACL'99)*, pages 269–270.

Chandrasekar, R., Christine Doran, and B. Srinivas. 1996. Motivations and methods for text simplification. In *Proceedings of the Sixteenth International Conference on Computational Linguistics (COLING'96)*, pages 1041–1044.

Filippova, Katja and Michael Strube. 2008a. Dependency tree based sentence compression. In *International Natural Language Generation Conference (INLG'08)*, pages 25–32.

Filippova, Katja and Michael Strube. 2008b. Sentence fusion via dependency graph compression. In *EMNLP '08: Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 177–185.

Graehl, Jonathan, Kevin Knight, and Jonathan May. 2008. Training tree transducers. In *Computational Linguistics*, volume 34, pages 391–427. MIT Press.

Heilman, M. and N. A. Smith. 2009. Question generation via overgenerating transformations and ranking. Technical Report CMU-LTI-09-013, Language Technologies Institute, Carnegie Mellon University.

Inui, Kentaro, Atsushi Fujita, Tetsuro Takahashi, Ryu Iida, and Tomoya Iwakura. 2003. Text simplification for reading assistance: A project note. In *Proceedings of the 2nd International Workshop on Paraphrasing: Paraphrase Acquisition and Applications (IWP)*, pages 9–16.

Jonnalagadda, Siddhartha and Graciela Gonzalez. 2009. Sentence simplification aids protein-protein interaction extraction. In *Proceedings of the 3rd International Symposium on Languages in Biology and Medicine*.

Jurafsky, Daniel and James H. Martin. 2008. *Speech and Language Processing (2nd Edition)*. Prentice Hall, 2 edition.

Klein, Dan and Christopher D. Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems 15 (NISP'02)*, pages 3–10.

Knight, Kevin and Daniel Marcu. 2000. Statistics-based summarization - step one: Sentence compression. In *AAAI*, pages 703–710.

Levenshtein. 1966. Binary code capable of correcting deletions, insertions and reversals. In *Soviet Physics*, pages 707–710.

Nelken, Rani and Stuart M. Shieber. 2006. Towards robust context-sensitive sentence alignment for monolingual corpora. In *Proceedings of 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 161–168.

Petersen, Sarah E. and Mari Ostendorf. 2007. Text simplification for language learners: a corpus analysis. In *Proc. of Workshop on Speech and Language Technology for Education*, pages 69–72.

Schmid, Helmut. 1994. Probabilistic part-of-speech tagging using decision trees. In *International Conference on New Methods in Language Processing*, pages 44–49.

Schwarm, Sarah E. and Mari Ostendorf. 2005. Reading level assessment using support vector machines and statistical language models. In *ACL'05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 523–530.

Siddharthan, Advaith. 2002. An architecture for a text simplification system. In *Proceedings of the Language Engineering Conference (LEC'02)*, pages 64–71.

Siddharthan, Advaith. 2006. Syntactic simplification and text cohesion. In *Research on Language & Computation*, volume 4, pages 77–109. Springer Netherlands, June.

Stolcke, Andreas. 2002. SRILM - An Extensible Language Modeling Toolkit. pages 901–904.

Vickrey, David and Daphne Koller. 2008. Sentence simplification for semantic role labeling. In *Proceedings of ACL-08: HLT*, pages 344–352, June.

Watanabe, Willian Massami, Arnaldo Candido Junior, Vinícius Rodriguez Uzêda, Renata Pontin de Mattos Fortes, Thiago Alexandre Salgueiro Pardo, and Sandra Maria Aluísio. 2009. Facilita: reading assistance for low-literacy readers. In *SIGDOC '09: Proceedings of the 27th ACM international conference on Design of communication*, pages 29–36. ACM.

Yamada, Kenji and Kevin Knight. 2001. A syntax-based statistical translation model. In *ACL'01: Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 523–530.

Yamada, Kenji and Kevin Knight. 2002. A decoder for syntax-based statistical mt. In *ACL'02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 303–310.

Zesch, Torsten, Christof Müller, and Iryna Gurevych. 2008. Extracting Lexical Semantic Knowledge from Wikipedia and Wiktionary. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, pages 1646–1652.

Zhao, Shiqi, Xiang Lan, Ting Liu, and Sheng Li. 2009. Application-driven statistical paraphrase generation. In *Proceedings of ACL-IJCNLP*, pages 834–842, Suntec, Singapore, August.

A Minimum Error Weighting Combination Strategy for Chinese Semantic Role Labeling

Tao Zhuang and **Chengqing Zong**
National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences
{tzhuang, cqzong}@nlpr.ia.ac.cn

Abstract

Many Semantic Role Labeling (SRL) combination strategies have been proposed and tested on English SRL task. But little is known about how much Chinese SRL can benefit from system combination. And existing combination strategies trust each individual system's output with the same confidence when merging them into a pool of candidates. In our approach, we assign different weights to different system outputs, and add a weighted merging stage to the conventional SRL combination architecture. We also propose a method to obtain an appropriate weight for each system's output by minimizing some error function on the development set. We have evaluated our strategy on Chinese Proposition Bank data set. With our minimum error weighting strategy, the F_1 score of the combined result achieves 80.45%, which is 1.12% higher than baseline combination method's result, and 4.90% higher than the best individual system's result.

1 Introduction

In recent years, Chinese Semantic Role Labeling has received much research effort (Sun and Jurafsky, 2004; Xue, 2008; Che et al., 2008; Ding and Chang, 2008; Sun et al., 2009; Li et al., 2009). And Chinese SRL is also included in CoNLL-2009 shared task (Hajič et al., 2009). On the data set used in (Xue, 2008), the F_1 score of the SRL results using automatic syntactic analysis is still in low 70s (Xue, 2008; Che et al., 2008; Sun et al., 2009). As pointed out by Xue (Xue, 2008), the SRL errors are mainly caused by the errors in automatic syntactic analysis. In fact, Chinese SRL suffers from parsing errors even more than English SRL, because the state-of-the-art parser for Chinese is still not as good as that for English. And previous research on English SRL shows that combination is a robust and effective method to alleviate SRL's dependency on parsing results (Màrquez et al., 2005; Koomen et al., 2005; Pradhan et al., 2005; Surdeanu et al., 2007; Toutanova et al., 2008). However, the effect of combination for Chinese SRL task is still unknown. This raises two questions at least: (1) How much can Chinese SRL benefit from combination? (2) Can existing combination strategies be improved? All existing combination strategies trust each individual system's output with the same confidence when putting them into a pool of candidates. But according to our intuition, different systems have different performance. And the system that have better performance should be trusted with more confidence. We can use our prior knowledge about the combined systems to do a better combination.

The observations above motivated the work in this paper. Instead of directly merging outputs with equal weights, different outputs are assigned different weights in our approach. An output's weight stands for the confidence we have in that output. We acquire these weights by minimizing an error function on the development set. And we use these weights to merge the outputs. In this paper, outputs are generated by a full parsing based Chinese SRL system and a shallow parsing based SRL system. The full parsing based system

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1362–1370,
Beijing, August 2010

use multiple parse trees to generate multiple SRL outputs. Whereas the shallow parsing based system only produce one SRL output. After merging all SRL outputs, we use greedy and integer linear programming combination methods to combine the merged outputs.

We have evaluated our combination strategy on Chinese Propbank data set used in (Xue, 2008) and get encouraging results. With our minimum error weighting (MEW) strategy, the F_1 score of the combined result achieves 80.45%. This is a significant improvement over the best reported SRL performance on this data set, which is 74.12% in the literature (Sun et al., 2009).

2 Related work

A lot of research has been done on SRL combination. Most of them focused on English SRL task. But the combination methods are general. And they are closely related to the work in this paper.

Punyakanok et al. (2004) formulated an Integer Linear Programming (ILP) model for SRL. Based on that work, Koomen et al. (2005) combined several SRL outputs using ILP method. Màrquez et al. (2005) proposed a combination strategy that does not require the individual system to give a score for each argument. They used a binary classifier to filter different systems' outputs. Then they used a greedy method to combine the candidates that pass the filtering process. Pradhan et al. (2005) combined systems that are based on phrase-structure parsing, dependency parsing, and shallow parsing. They also used greedy method when combining different outputs. Surdeanu et al. (2007) did a complete research on a variety of combination strategies. All these research shows that combination can improve English SRL performance by 2$\sim$5 points on F_1 score. However, little is known about how much Chinese SRL can benefit from combination. And, as we will show, existing combination strategies can still be improved.

3 Individual SRL Systems

3.1 Full Parsing Based System

The full parsing based system utilize full syntactic analysis to perform semantic role labeling. We implemented a Chinese semantic role labeling system similar to the one described in (Xue, 2008). Our system consists of an argument identification stage and an argument classification stage. In the argument identification stage, a number of argument locations are identified in a sentence. In the argument classification stage, each location identified in the first stage is assigned a semantic role label. The features used in this paper are the same with those used in (Xue, 2008).

Maximum entropy classifier is employed for both the argument identification and classification tasks. And Zhang Le's MaxEnt toolkit[1] is used for implementation.

3.2 Shallow Parsing Based System

The shallow parsing based system utilize shallow syntactic information at the level of phrase chunks to perform semantic role labeling. Sun et al. (2009) proposed such a system on Chinese SRL and reported encouraging results. The system used in this paper is based on their approach. For Chinese chunking, we adopted the method used in (Chen et al., 2006), in which chunking is regarded as a sequence labeling task with IBO2 representation. The features used for chunking are the uni-gram and bi-gram word/POS tags with a window of size 2. The SRL task is also regarded as a sequence labeling problem. For an argument with label ARG*, we assign the label B-ARG* to its first chunk, and the label I-ARG* to its rest chunks. The chunks outside of any argument are assigned the label O. The features used for SRL are the same with those used in the one-stage method in (Sun et al., 2009).

In this paper, we employ Tiny SVM along with Yamcha (Kudo and Matsumoto, 2001) for Chinese chunking, and CRF++[2] for SRL.

3.3 Individual systems' outputs

The maximum entropy classifier used in full parsing based system and the CRF model used in shallow paring based system can both output classification probabilities. For the full parsing based system, the classification probability of the ar-

[1] http://homepages.inf.ed.ac.uk/lzhang10/maxent_toolkit.html

[2] http://crfpp.sourceforge.net/

gument classification stage is used as the argument's probability. Whereas for the shallow parsing based system, an argument is usually comprised of multiple chunks. For example, an argument with label ARG0 may contain three chunks labeled as: B-ARG0, I-ARG0, I-ARG0. And each chunk has a label probability. Thus we have three probabilities p_1, p_2, p_3 for one argument. In this case, we use the geometric mean of individual chunks' probabilities $(p_1 \cdot p_2 \cdot p_3)^{1/3}$ as the argument's probability.

As illustrated in Figure 1, in an individual system's output, each argument has three attributes: its location in sentence loc, represented by the number of its first word and last word; its semantic role label l; and its probability p.

Sent:	外商	投资 企业	成为 中国 外贸 重要 增长点	
Args:	[ARG0]	[pred]	[ARG1	]
loc:	(0, 2)		(4, 7)	
l:	ARG0		ARG1	
p:	0.94		0.92	

Figure 1: Three attributes of an output argument: location loc, label l, and probability p.

So each argument outputted by a system is a triple (loc, l, p). For example, the ARG0 in Figure 1 is $((0, 2), \text{ARG0}, 0.94)$. Because the outputs of baseline systems are to be combined, we call such triple a **candidate** for combination.

4 Approach Overview

As illustrated in Figure 2, the architecture of our system consists of a candidates generation stage, a weighted merging stage, and a combination stage. In the candidates generation stage, the baseline systems are run individually and their outputs are collected. We use 2-best parse trees of Berkeley parser (Petrov and Klein, 2007) and 1-best parse tree of Bikel parser (Bikel, 2004) and Stanford parser (Klein and Manning, 2003) as inputs to the full parsing based system. The second best parse tree of Berkeley parser is used here for its good quality. So together we have four different outputs from the full parsing based system. From the shallow parsing based system, we have only one output.

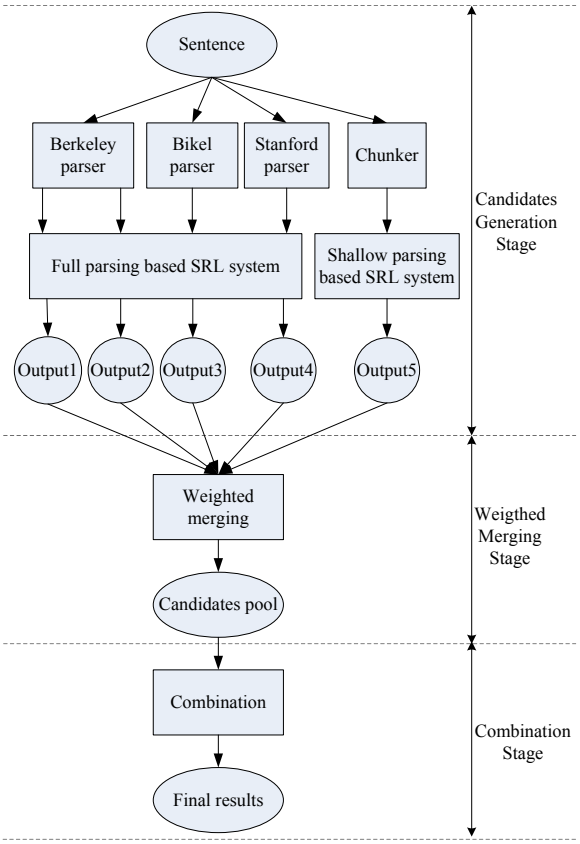


Figure 2: The overall architecture of our system.

In the weighted merging stage, each system output is assigned a weight according to our prior knowledge obtained on the development set. Details about how to obtain appropriate weights will be explained in Section 6. Then all candidates with the same loc and l are merged to one by weighted summing their probabilities. Specifically, suppose that there are n system outputs to be combined, with the i-th output's weight to be w_i. And the candidate in the i-th output with loc and l is (loc, l, p_i) (If there is no candidate with loc and l in the i-th output, p_i is 0.). Then the merged candidate is (loc, l, p), where $p = \sum_{i=1}^{n} w_i p_i$.

After the merging stage, a pool of merged candidates is obtained. In the combination stage, candidates in the pool are combined to form a consistent SRL result. Greedy and integer linear programming combination methods are experimented in this paper.

5 Combination Methods

5.1 Global constraints

When combining the outputs, two global constraints are enforced to resolve the conflict between outputs. These two constraints are:

1. *No duplication*: There is no duplication for key arguments: ARG0 $\sim$ ARG5.

2. *No overlapping*: Arguments cannot overlap with each other.

We say two argument candidates **conflict** with each other if they do not satisfy the two constraints above.

5.2 Two combination methods

Under these constraints, two methods are explored to combine the outputs. The first one is a greedy method. In this method, candidates with probability below a threshold are deleted at first. Then the remaining candidates are inspected in descending order according to their probabilities. And each candidate will be put into a solution set if it does not conflict with candidates already in the set. This greedy combination method is very simple and has been adopted in previous research (Pradhan et al., 2005; Màrquez et al., 2005).

The second combination method is integer linear programming (ILP) method. ILP method was first applied to SRL in (Punyakanok et al., 2004). Here we formulate an ILP model whose form is different from the model in (Punyakanok et al., 2004; Koomen et al., 2005). For convenience, we denote the whole label set as $\{l_1, l_2, \ldots, l_n\}$. And let $l_1 \sim l_6$ stand for the key argument labels ARG0 $\sim$ ARG5 respectively. Suppose there are m different locations, denoted as $loc_1, \ldots, loc_m$, among all candidates in the pool. And the probability of assigning l_j to loc_i is p_{ij}. A binary variable x_{ij} is defined as:

$$x_{ij} = \begin{cases} 1 & \text{if } loc_i \text{ is assigned label } l_j, \\ 0 & \text{otherwise.} \end{cases}$$

The objective of the ILP model is to maximize the sum of arguments' probabilities:

$$\max \sum_{i=1}^{m} \sum_{j=1}^{n} (p_{ij} - T) x_{ij} \qquad (1)$$

where T is a threshold to prevent including too many candidates in solution. T is similar to the threshold in greedy combination method. In this paper, both thresholds are empirically tuned on development data, and both are set to be 0.2.

The inequalities in equation (2) make sure that each loc is assigned at most one label.

$$\forall 1 \leq i \leq m : \sum_{j=1}^{n} x_{ij} \leq 1 \qquad (2)$$

The inequalities in equation (3) satisfy the *No duplication* constraint.

$$\forall 1 \leq j \leq 6 : \sum_{i=1}^{m} x_{ij} \leq 1 \qquad (3)$$

For any location loc_i, let C_i denote the index set of the locations that overlap with it. Then the *No overlapping* constraint means that if loc_i is assigned a label, i.e., $\sum_{j=1}^{n} x_{ij} = 1$, then for any $k \in C_i$, loc_k cannot be assigned any label, i.e., $\sum_{j=1}^{n} x_{kj} = 0$. A common technique in ILP modeling to form such a constraint is to use a sufficiently large auxiliary constant M. And the constraint is formulated as:

$$\forall 1 \leq i \leq m : \sum_{k \in C_i} \sum_{j=1}^{n} x_{kj} \leq (1 - \sum_{j=1}^{n} x_{ij}) M \qquad (4)$$

In this case, M only needs to be larger than the number of candidates to be combined. In this paper, $M = 500$ is large enough. And we employ lpsolve[3] to solve the ILP model.

Note that the form of the ILP model in this paper is different from that in (Punyakanok et al., 2004; Koomen et al., 2005) in three aspects: (1) A special label class *null*, which means no label is assigned, was added to the label set in (Punyakanok et al., 2004; Koomen et al., 2005). Whereas no such special class is needed in our model, because if no label is assigned to loc_i, $\sum_{j=1}^{n} x_{ij} = 0$ would simply indicate this case. This makes our model contain fewer variables. (2) Without *null* class in our model, we need to use a different technique to formulate the *No-overlapping* constraint. (3) In order to compare

[3] http://lpsolve.sourceforge.net/

with the greedy combination method, the ILP model in this paper conforms to exactly the same constraints as the greedy method. Whereas many more global constraints were taken into account in (Punyakanok et al., 2004; Koomen et al., 2005).

6 Train Minimum Error Weights

The idea of minimum error weighting is straightforward. Individual outputs $O_1, O_2, \ldots, O_n$ are assigned weights $w_1, w_2, \ldots, w_n$ respectively. These weights are normalized, i.e., $\sum_{i=1}^{n} w_i = 1$. An output's weight can be seen as the confidence we have in that output. It is a kind of prior knowledge we have about that output. We can gain this prior knowledge on the development set. As long as the data of the development set and the test set are similar, this prior knowledge should be able to help to guide SRL combination on test set. In this section, we discuss how to obtain appropriate weights.

6.1 Training model

Suppose the golden answer and SRL result on development set are d and r respectively. An **error function** $Er(r, d)$ is a function that measures the error contained in r in reference to d. An error function can be defined as the number of wrong arguments in r. It can also be defined using precision, recall, or F_1 score. For example, $Er(r, d) = 1 - Precision(r, d)$, or $Er(r, d) = 1 - F_1(r, d)$. Smaller value of error function means less error in r.

The combination process can also be seen as a function, which maps the outputs and weights to the combined result r: $r = Comb(O_1^n, w_1^n)$. Therefore, the error function of our system on development set is:

$$Er(r, d) = Er(Comb(O_1^n, w_1^n), d) \qquad (5)$$

From equation (5), it can be seen that: Given development set d, if the outputs to be combined O_1^n and the combination method $Comb$ are fixed, the error function is just a function of the weights. So we can obtain appropriate weights by minimizing the error function:

$$\hat{w}_1^n = \arg\min_{w_1^n} Er(Comb(O_1^n, w_1^n), d) \qquad (6)$$

6.2 Training algorithm

Algorithm 1 Powell Training Algorithm.

1: **Input** : Error function $Er(\boldsymbol{w})$.
2: Initialize n directions $\boldsymbol{d}_1, \ldots, \boldsymbol{d}_n$, and a start point $\boldsymbol{w}$ in R^n.
3: Set termination threshold δ.
4: **do**:
5: $\quad \boldsymbol{w}_1 \leftarrow \boldsymbol{w}$
6: $\quad$ **for** $i \leftarrow 1, \ldots, n$:
7: $\quad\quad \alpha_i \leftarrow \arg\min_{\alpha} f(\boldsymbol{w}_i + \alpha \boldsymbol{d}_i)$
8: $\quad\quad \boldsymbol{w}_{i+1} \leftarrow \boldsymbol{w}_i + \alpha_i \boldsymbol{d}_i$
9: $\quad \boldsymbol{d}_{n+1} \leftarrow \boldsymbol{w}_{n+1} - \boldsymbol{w}$
10: $\quad \alpha^* \leftarrow \arg\min_{\alpha} f(\boldsymbol{w} + \alpha \boldsymbol{d}_{n+1})$
11: $\quad \boldsymbol{w}' \leftarrow \boldsymbol{w} + \alpha^* \boldsymbol{d}_{n+1}$
12: $\quad \Delta Er \leftarrow Er(\boldsymbol{w}) - Er(\boldsymbol{w}')$
13: $\quad i \leftarrow \arg\max_{1 \le j \le n} Er(\boldsymbol{w}_j) - Er(\boldsymbol{w}_{j+1})$
14: $\quad$ **if** $(\alpha^*)^2 \ge \frac{\Delta Er}{Er(\boldsymbol{w}_i) - Er(\boldsymbol{w}_{i+1})}$:
15: $\quad\quad$ **for** $j \leftarrow i, \ldots, n$:
16: $\quad\quad\quad \boldsymbol{d}_j \leftarrow \boldsymbol{d}_{j+1}$
17: $\quad \boldsymbol{w} \leftarrow \boldsymbol{w}'$
18: **while** $\Delta Er > \delta$
19: **Output**: The minimum error weights $\boldsymbol{w}$.

There are two difficulties to solve the optimization problem in equation 6. The first one is that the error function cannot be written to an analytical form. This is because the $Comb$ function, which stands for the combination process, cannot be written as an analytical formula. So the problem cannot be solved using canonical gradient-based optimization algorithms, because the gradient function cannot be derived. The second difficulty is that, according to our experience, the error function has many local optima, which makes it difficult to find a global optima.

To resolve the first difficulty, Modified Powell's method (Yuan, 1993) is employed to solve the optimization problem. Powell's method is a heuristic search method that does not require the objective function to have an explicit analytical form. The training algorithm is presented in Algorithm 1. In Algorithm 1, the line search problem in steps 7 and 10 is solved using Brent's method (Yuan, 1993). And the temination threshold δ is empirically set to be 0.001 in this paper.

To resolve the second difficulty, we perform multiple searches using different start points, and then choose the best solution found.

7 Experiments

7.1 Experimental setup

We use Chinese Proposition Bank (CPB) 1.0 and Chinese Tree Bank (CTB) 5.0 of Linguistic Data Consortium corpus in our experiments. The training set is comprised of 648 files(chtb_081.fid to chtb_885.fid). The development set is comprised of 40 files(chtb_041.fid to chtb_080.fid). The test set is comprised of 72 files(chtb_001.fid to chtb_040.fid and chtb_900.fid to chtb_931.fid).

The same data setting has been used in (Xue, 2008; Ding and Chang, 2008; Sun et al., 2009). Sun et al. (2009) used sentences with golden segmentation and POS tags as input to their SRL system. However, we use sentences with only golden segmentation as input. Then we perform automatic POS tagging using Stanford POS tagger (Toutanova et al., 2003). In (Xue, 2008), the parser used by the SRL system is trained on the training and development set plus 275K words of broadcast news. In this paper, all parsers used by the full parsing based system are trained on the training set plus the broadcast news portion of CTB6.0. And the chunker used in the shallow parsing based system is trained just on the training set.

7.2 Individual outputs' performance

In this paper the four outputs of the full parsing based system are represented by FO1 $\sim$ FO4 respectively. Among them, FO1 and FO2 are the outputs using the first and second best parse trees of Berkeley parser, FO3 and FO4 are the outputs using the best parse trees of Stanford parser and Bikel parser respectively. The output of the shallow parsing based system is represented by SO. The individual outputs' performance on development and test set are listed in Table 1.

From Table 1 we can see that the performance of individual outputs are similar on development set and test set. On both sets, the F_1 scores of individual outputs are in the same order: FO1 $>$ FO2 $>$ SO $>$ FO3 $>$ FO4.

Data set	Outputs	$P(\%)$	$R(\%)$	F_1
development	FO1	**79.17**	**72.09**	**75.47**
	FO2	77.89	70.56	74.04
	FO3	72.57	67.02	69.68
	FO4	75.60	63.45	69.00
	SO	73.72	67.35	70.39
test	FO1	**80.75**	**70.98**	**75.55**
	FO2	79.44	69.37	74.06
	FO3	73.95	66.37	70.00
	FO4	75.89	63.26	69.00
	SO	75.69	67.90	71.59

Table 1: The results of individual systems on development and test set.

7.3 Combining outputs of full parsing based system

In order to investigate the benefit that the full parsing based system can get from using multiple parsers, we combine the four outputs FO1 $\sim$ FO4. The combination results are listed in Table 2. In tables of this paper, "Grd" and "ILP" stand for greedy and ILP combination methods respectively, and "+MEW" means the combination is performed with MEW strategy.

	$P(\%)$	$R(\%)$	F_1
Grd	**82.68**	73.36	77.74
ILP	82.21	73.93	77.85
Grd+MEW	81.30	75.38	78.23
ILP+MEW	81.27	**75.74**	**78.41**

Table 2: The results of combining outputs of full parsing based system on test set.

	Er	FO1	FO2	FO3	FO4
Grd	$1-F_1$	0.31	0.16	0.30	0.23
ILP	$1-F_1$	0.33	0.10	0.27	0.30

Table 3: The minimum error weights for the results in Table 2.

From Table 2 and Table 1, we can see that, without MEW strategy, the F_1 score of combination result is about 2.3% higher than the best individual output. With MEW strategy, the F_1 score is improved about 0.5% further. That is to say, with MEW strategy, the benefit of combination is improved by about 20%. Therefore, the effect of MEW is very encouraging.

Here the error function for MEW training is chosen to be $1 - F_1$. And the trained weights for greedy and ILP methods are listed in Table 3

separately. In tables of this paper, the column Er corresponds to the error function used for MEW strategy.

7.4 Combining all outputs

We have also combined all five outputs. The results are listed in Table 4. Compared with the results in Table 2, we can see that the combination results is largely improved, especially the recall.

	$P(\%)$	$R(\%)$	F_1
Grd	**83.64**	75.32	79.26
ILP	83.31	75.71	79.33
Grd+MEW	83.34	77.47	80.30
ILP+MEW	83.02	**78.03**	**80.45**

Table 4: The results of combining all outputs on test set.

From Table 4 and Table 1 we can see that without MEW strategy, the F_1 score of combination result is about 3.8% higher than the best individual output. With MEW, the F_1 score is improved further by more than 1%. That means the benefit of combination is improved by over 25% with MEW strategy.

Here the error function for MEW training is still $1 - F_1$, and the trained weights are listed in Table 5.

	Er	FO1	FO2	FO3	FO4	SO
Grd	$1 - F_1$	0.23	0.12	0.23	0.20	0.22
ILP	$1 - F_1$	0.24	0.08	0.22	0.21	0.25

Table 5: The minimum error weights for the results in Table 4.

7.5 Using alternative error functions for minimum error weights training

In previous experiments, we use $1 - F_1$ as error function. As pointed out in Section 6, the definition of error function is very general. So we have experimented with two other error functions, which are $1 - Precision$, and $1 - Recall$. Obviously, these two error functions favor precision and recall separately. The results of combining all five outputs using these two error functions are listed in Table 6, and the trained weights are listed in Table 7.

From Table 6 and Table 4, we can see that when $1 - Precison$ is used as error function, the precision of combination result is largely improved. But the recall decreases a lot. Similar effect of the error function $1 - Recall$ is also observed.

	Er	$P(\%)$	$R(\%)$	F_1
Grd+MEW	$1 - P$	85.31	73.42	78.92
ILP+MEW	$1 - P$	**85.62**	72.76	78.67
Grd+MEW	$1 - R$	81.94	77.55	79.68
ILP+MEW	$1 - R$	79.74	**78.34**	79.03

Table 6: The results of combining all outputs with alternative error functions.

	Er	FO1	FO2	FO3	FO4	SO
Grd	$1 - P$	0.25	0.24	0.22	0.22	0.07
ILP	$1 - P$	0.30	0.26	0.20	0.15	0.09
Grd	$1 - R$	0.21	0.10	0.17	0.15	0.37
ILP	$1 - R$	0.24	0.04	0.10	0.22	0.39

Table 7: The minimum error weights for the results in Table 6.

The results of this subsection reflect the flexibility of MEW strategy. This flexibility comes from the generality of the definition of error function. The choice of error function gives us some control over the results we want to get. We can define different error functions to favor precision, or recall, or some error counts such as the number of misclassified arguments.

7.6 Discussion

In this paper, the greedy and ILP combination methods conform to the same simple constraints specified in Section 5. From the experiment results, we can see that ILP method generates slightly better results than greedy method.

In Subsection 7.4, we see that combining all outputs using ILP method with MEW strategy yields 4.90% improvement on F_1 score over the best individual output FO1. In order to understand each output's contribution to the improvement over FO1. We compare the differences between outputs.

Let C_O denote the set of correct arguments in an output O. Then we get the following statistics when comparing two outputs A and B: (1) the number of common correct arguments in A and B, i.e., $|C_A \cap C_B|$; (2) the number of correct arguments in A and not in B, i.e., $|C_A \setminus C_B|$; (3) the number of correct arguments in B and not in A, i.e., $|C_B \setminus C_A|$. The comparison results between

some outputs on test set are listed in Table 8. In this table, UF stands for the union of the 4 outputs FO1 $\sim$ FO4.

A	B	$\lvert C_A \cap C_B \rvert$	$\lvert C_A \setminus C_B \rvert$	$\lvert C_B \setminus C_A \rvert$
FO1	FO2	5498	508	372
	FO3	5044	962	552
	FO4	4815	1191	512
	SO	4826	1180	920
UF	SO	5311	1550	435

Table 8: Comparison between outputs on test set.

From Table 8 we can see that the output SO has 4826 common correct arguments with FO1, which is relatively small. And, more importantly, SO contains 920 correct arguments not in FO1, which is much more than any other output contains. Therefore, SO is more complementary to FO1 than other outputs. On the contrary, FO2 is least complementary to FO1. Even compared with the union of FO1 $\sim$ FO4, SO still contains 435 correct arguments not in the union. This shows that the output of shallow parsing based system is a good complement to the outputs of full parsing based system. This explains why recall is largely improved when SO is combined in Subsection 7.4. From the analysis above we can also see that the weights in Table 5 are quite reasonable. In Table 5, SO is assigned the largest weight and FO2 is assigned the smallest weight.

In Subsection 7.3, the MEW strategy improves the benefit of combination by about 20%. And in Subsection 7.4, the MEW strategy improves the benefit of combination by over 25%. This shows that the MEW strategy is very effective for Chinese SRL combination.

To our best knowledge, no results on Chinese SRL combination has been reported in the literature. Therefore, to compare with previous results, the top two results of single SRL system in the literature and the result of our combination system on this data set are listed in Table 9. For the results in Table 9, the system of Sun et al. uses sentences with golden POS tags as input. Xue's system and our system both use sentences with automatic POS tags as input. The result of Sun et al. (2009) is the best reported result on this data set in the literature.

	POS	$P(\%)$	$R(\%)$	F_1
(Xue, 2008)	auto	76.8	62.5	68.9
(Sun et al., 2009)	gold	79.25	69.61	74.12
Ours	auto	**83.02**	**78.03**	**80.45**

Table 9: Previous best single system's results and our combination system's result on this data set.

8 Conclusions

In this paper, we propose a minimum error weighting strategy for SRL combination and investigate the benefit that Chinese SRL can get from combination. We assign different weights to different system outputs and add a weighted merging stage to conventional SRL combination system architecture. And we also propose a method to train these weights on development set. We evaluate the MEW strategy on Chinese Propbank data set with greedy and ILP combination methods.

Our experiments have shown that the MEW strategy is very effective for Chinese SRL combination, and the benefit of combination can be improved over 25% with this strategy. And also, the MEW strategy is very flexible. With different definitions of error function, this strategy can favor precision, or recall, or F_1 score. The experiments have also shown that Chinese SRL can benefit a lot from combination, especially when systems based on different syntactic views are combined. The SRL result with the highest F_1 score in this paper is generated by ILP combination together with MEW strategy. In fact, the MEW strategy is easy to incorporate with other combination methods, just like incorporating with the greedy and ILP combination methods in this paper.

Acknowledgment

The research work has been partially funded by the Natural Science Foundation of China under Grant No. 60975053, 90820303 and 60736014, the National Key Technology R&D Program under Grant No. 2006BAH03B02, the Hi-Tech Research and Development Program ("863" Program) of China under Grant No. 2006AA010108-4, and also supported by the China-Singapore Institute of Digital Media (CSIDM) project under grant No. CSIDM-200804.

References

Daniel Bikel. 2004. Intricacies of Collins Parsing Model. *Computational Linguistics*, 30(4):480-511.

Wanxiang Che, Min Zhang, Ai Ti Aw, Chew Lim Tan, Ting Liu, and Sheng Li. 2008. Using a Hybrid Convolution Tree Kernel for Semantic Role Labeling. *ACM Transactions on Asian Language Information Processing*, 2008, 7(4).

Wenliang Chen, Yujie Zhang, and Hitoshi Isahara. 2006. An empirical study of Chinese chunking. In *Proceedings of COLING/ACL-2006*.

Weiwei Ding and Baobao Chang. 2008. Improving Chinese Semantic Role Classification with Hierarchical Feature Selection Strategy. In *Proceedings of EMNLP-2008*.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue and Yi Zhang. 2009. The CoNLL-2009 Shared Task: Syntactic and Semantic Dependencies in Multiple Languages. In *Proceedings of CoNLL-2009*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL-2003*.

Peter Koomen, Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2005. Generalized Inference with Multiple Semantic Role Labeling Systems. In *Proceedings of CoNLL-2005 shared task*.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with Support Vector Machines. In *Proceedings of NAACL-2001*.

Junhui Li, Guodong Zhou, Hai Zhao, Qiaoming Zhu, and Peide Qian. 2009. Improving Nominal SRL in Chinese Language with Verbal SRL Information and Automatic Predicate Recognition. In *Proceedings of EMNLP-2009*.

Lluís Màrquez, Mihai Surdeanu, Pere Comas, and Jordi Turmo. 2005. A Robust Combination Strategy for Semantic Role Labeling. In *Proceedings of EMNLP-2005*.

Slav Petrov and Dan Klein. 2007. Improved Inference for Unlexicalized parsing. In *Proceedings of ACL-2007*.

Sameer S. Pradhan, Wayne Ward, Kadri Hacioglu, James H. Martin, and Daniel Jurafsky. 2005. Semantic Role Labeling Using Different Syntactic Views. In *Proceedings of ACL-2005*.

Vasin Punyakanok, Dan Roth, Wen-tau Yih, and Dav Zimak. 2004. Semantic Role Labeling via Integer Linear Programming Inference. In *Proceedings of COLING-2004*.

Honglin Sun and Daniel Jurafsky. 2004. Shallow semantic parsing of Chinese. In *Proceedings of NAACL-2004*.

Weiwei Sun, Zhifang Sui, Meng Wang, and Xin Wang. 2009. Chinese Semantic Role Labeling with Shallow Parsing. In *Proceedings of EMNLP-2009*.

Mihai Surdeanu, Lluís Màrquez, Xavier Carreras, and Pere R. Comas. 2007. Combination Strategies for Semantic Role Labeling. *Journal of Artificial Intelligence Research (JAIR)*, 29:105-151.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2008. A Global Joint Model for Semantic Role Labeling. *Computational Linguistics*, 34(2): 145-159.

Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *Proceedings of HLT-NAACL-2003*.

Nianwen Xue. 2008. Labeling Chinese Predicates with Semantic Roles. *Computational Linguistics*, 34(2): 225-255.

Yaxiang Yuan. 1993. *Numerical Methods for Nonlinear Programming*. Shanghai Scientific and Technical Pulishers, Shanghai.

Detecting Speech Repairs Incrementally Using a Noisy Channel Approach

Simon Zwarts, Mark Johnson and Robert Dale
Centre for Language Technology
Macquarie University
{simon.zwarts|mark.johnson|robert.dale}@mq.edu.au

Abstract

Unrehearsed spoken language often contains disfluencies. In order to correctly interpret a spoken utterance, any such disfluencies must be identified and removed or otherwise dealt with. Operating on transcripts of speech which contain disfluencies, our particular focus here is the identification and correction of speech repairs using a noisy channel model. Our aim is to develop a high-accuracy mechanism that can identify speech repairs in an incremental fashion, as the utterance is processed word-by-word.

We also address the issue of the evaluation of such incremental systems. We propose a novel approach to evaluation, which evaluates performance in detecting and correcting disfluencies incrementally, rather than only assessing performance once the processing of an utterance is complete. This demonstrates some shortcomings in our basic incremental model, and so we then demonstrate a technique that improves performance on the detection of disfluencies as they happen.

1 Introduction

One of the most obvious differences between written language and spoken language is the fact that the latter presents itself incrementally over some time period. Most natural language processing applications operate on complete sentences; but for real time spontaneous speech, there are potential benefits to incrementally processing the input so that a system can stay responsive and interact directly before a speaker's utterance is complete. Work in psycholinguistics supports the view that the human parsing mechanism works incrementally, with partial semantic interpretations being produced before the complete utterance has been heard (Marslen-Wilson, 1973). Our interest is in developing similarly incremental processing techniques for natural language interpretation, so that, for example, a speech recognizer might be able to interject during a long utterance to object, cut the speaker short, or correct a mistaken assumption; such a mechanism is even required for the appropriate timing of backchannel signals. Additionally the incremental nature of the model allows potential application of this model in speech recognition models.

Another feature of unrehearsed spoken language that has no obvious correlate in written language is the presence of disfluencies.[1] Disfluencies are of different types, ranging from simple filled pauses (such as *um* and *uh*) to more complicated structures where the sequence of words that make up the utterance is 'repaired' while it is being produced. Whereas simpler disfluencies may be handled by simply deleting them from the sequence of words under consideration, the editing terms in a speech repair are part of the utterance, and therefore require more sophisticated processing.

There are three innovations in the present paper. First, we demonstrate that a noisy channel model of speech repairs can work accurately in an incremental fashion. Second, we provide an approach to the evaluation of

[1] Although some disfluencies can be considered grammatical errors, they are generally quite distinct in both cause and nature from the kinds of grammatical errors found in written text.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1371–1378,
Beijing, August 2010

such an incremental model. Third, we tackle the problem of the early detection of speech repairs, and demonstrate a technique that decreases the latency (as measured in tokens) involved in spotting that a disfluency has occurred.

The rest of the paper is structured as follows. Section 2 provides some background on speech repairs and existing approaches to handling them, including Johnson and Charniak's (2004) model, which we use as a starting point for our incremental model. Section 3 describes our model in detail, focusing on the noisy channel model and the incremental component of this model. Section 4 introduces some considerations that arise in the development of techniques for the evaluation of incremental disfluency detection; we then provide a quantitative assessment of our performance using these techniques. Our evaluation reveals that our basic incremental model does not perform very well at detecting disfluencies close to where they happen, so in Section 5 we present a novel approach to optimise detection of these disfluencies as early as possible. Finally Section 6 concludes and discusses future work.

2 Speech Repairs

We adopt the terminology and definitions introduced by Shriberg (1994) to discuss disfluency. We are particularly interested in what are called **repairs**. These are the hardest types of disfluency to identify since they are not marked by a characteristic vocabulary. Shriberg (1994) identifies and defines three distinct parts of a repair, referred to as the **reparandum**, the **interregnum** and the **repair**. Consider the following utterance:

$$\underbrace{\textit{uh, I mean}}_{\text{interregnum}}\ \underbrace{\textit{to Denver}}_{\text{repair}}\ \textit{on Friday} \qquad \textit{I want a flight}\ \overbrace{\textit{to Boston,}}^{\text{reparandum}} \tag{1}$$

The reparandum *to Boston* is the part of the utterance that is being edited out; the interregnum *uh* is a filler, which may not always be present; and the repair *to Denver* replaces the reparandum.

Given an utterance that contains such a repair, we want to be able to correctly detect the start and end positions of each of these three components. We can think of each word in an utterance as belonging to one of four categories: fluent material, reparandum, interregnum, or repair. We can then assess the accuracy of techniques that attempt to detect disfluencies by computing precision and recall values for the assignment of the correct categories to each of the words in the utterance, as compared to the gold standard as indicated by annotations in the corpus.

An alternative means of evaluation would be to simply generate a new signal with the reparandum and filler removed, and compare this against a 'cleaned-up' version of the utterance; however, Core and Schubert (1999) argue that, especially in the case of speech repairs, it is important not to simply throw away the disfluent elements of an utterance, since they can carry meaning that needs to be recovered for proper interpretation of the utterance. We are therefore interested in the first instance in a model of speech error *detection*, rather than a model of *correction*.

Johnson and Charniak (2004) describe such a model, using a noisy-channel based approach to the detection of the start and end points of reparanda, interregna and repairs. Since we use this model as our starting point, we provide a more detailed explanation in Section 3.

The idea of using a noisy channel model to identify speech repairs has been explored for languages other than English. Honal and Schultz (2003) use such a model, comparing speech disfluency detection in spontaneous spoken Mandarin against that in English. The approach performs well in Mandarin, although better still in English.

Both the models just described operate on transcripts of completed utterances. Ideally, however, when we deal with speech we would like to process the input word by word as it is received. Being able to do this would enable tighter integration in both speech recognition

and interpretation, which might in turn improve overall accuracy.

The requirement for incrementality is recognised by Schuler et al. (2010), who employ an incremental Hierarchical Hidden Markov Model (HHMM) to detect speech disfluencies. The HHMM is trained on manually annotated parse trees which are transformed by a right corner transformation; the HHMM is then used in an incremental fashion on unseen data, growing the parse structure each time a new token comes in. Special subtrees in this parse can carry a marker indicating that the span of the subtree consists of tokens corresponding to a speech disfluency. Schuler et al.'s approach thus provides scope for detecting disfluencies in an incremental fashion. However, their reported accuracy scores are not as good as those of Johnson and Charniak (2004): they report an F-score of 0.690 for their HHMM+RCT model, as compared to 0.797 for Johnson and Charniak's parser model.

Our aim in this paper, then, is to investigate whether it is possible to adapt Johnson and Charniak's model to process utterances incrementally, without any loss of accuracy. To define the incremental component more precisely, we investigate the possibility of marking the disfluencies as soon as possible during the processing of the input. Given two models that provide comparable accuracy measured on utterance completion, we would prefer a model which detects disfluencies earlier.

3 The Model

In this section, we describe Johnson and Charniak's (2004) noisy channel model, and show how this model can be made incremental.

As a data set to work with, we use the Switchboard part of the Penn Treebank 3 corpus. The Switchboard corpus is a corpus of spontaneous conversations between two parties. In Penn Treebank 3, the disfluencies are manually annotated. Following Johnson and Charniak (2004), we use all of sections 2 and 3 for training; we use conversations 4[5-9]* for a held-out training set; and conversations 40*, 41[0-4]* and 415[0-3]* as the held-out test set.

3.1 The Noisy Channel Model

To find the repair disfluencies a noisy channel model is used. For an observed utterance with disfluencies, y, we wish to find the most likely source utterance, $\hat{x}$, where:

$$\begin{aligned} \hat{x} &= argmax_x\ p(x \mid y) \\ &= argmax_x\ p(y \mid x)\, p(x) \end{aligned} \tag{2}$$

Here we have a channel model $p(y|x)$ which generates an utterance y given a source x and a language model $p(x)$. We assume that x is a substring of y, i.e., the source utterance can be obtained by marking words in y as a disfluency and effectively removing them from this utterance.

Johnson and Charniak (2004) experiment with variations on the language model; they report results for a bigram model, a trigram model, and a language model using the Charniak Parser (Charniak, 2001). Their parser model outperforms the bigram model by 5%. The channel model is based on the intuition that a reparandum and a repair are generally very alike: a repair is often almost a copy of the reparandum. In the training data, over 60% of the words in a reparandum are lexically identical to the words in the repair. Example 1 provides an example of this: half of the repair is lexically identical to the reparandum. The channel model therefore gives the highest probability when the reparandum and repair are lexically equivalent. When the potential reparandum and potential repair are not identical, the channel model performs deletion, insertion or substitution. The probabilities for these operations are defined on a lexical level and are derived from the training data. This channel model is formalised using a Synchronous Tree Adjoining Grammar (STAG) (Shieber and Schabes, 1990), which matches words from the reparandum to the repair. The weights for these STAG rules are learnt from the training text, where reparanda and repairs are aligned to each other using a minimum edit-distance string aligner.

For a given utterance, every possible utterance position might be the start of a reparandum, and every given utterance position thereafter might be the start of a repair (to limit complexity, a maximum distance between these two points is imposed). Every disfluency in turn can have an arbitrary length (again up to some maximum to limit complexity). After every possible disfluency other new reparanda and repairs might occur; the model does not attempt to generate crossing or nested disfluencies, although they do very occasionally occur in practice. To find the optimal selection for reparanda and repairs, all possibilities are calculated and the one with the highest probability is selected. A chart is filled with all the possible start and end positions of reparanda, interregna and repairs; each entry consists of a tuple $\langle rm_{\text{begin}}, ir_{\text{begin}}, rr_{\text{begin}}, rr_{\text{end}} \rangle$, where rm is the reparandum, ir is the interregnum and rr is the repair. A Viterbi algorithm is used to find the optimal path through the utterance, ranking each chart entry using the language model and channel model. The language model, a bigram model, can be easily calculated given the start and end positions of all disfluency components. The channel model is slightly more complicated because an optimal alignment between reparandum and repair needs to be calculated. This is done by extending each partial analysis by adding a word to the reparandum, the repair or both. The start position and end position of the reparandum and repair are given for this particular entry. The task of the channel model is to calculate the highest probable alignment between reparandum and repair. This is done by initialising with an empty reparandum and repair, and 'growing' the analysis one word at a time. Using a similar approach to that used in calculating the edit-distance between reparandum and repair, the reparandum and repair can both be extended with one of four operations: deletion (only the reparandum grows), insertion (only the repair grows), substitution (both grow), or copy (both grow). When the reparandum and the repair have their length corresponding to the current entry in the chart, the channel probability can be calculated. Since there are multiple alignment possibilities, we use dynamic programming to select the most probable solutions. The probabilities for insertion, deletion or substitution are estimated from the training corpus. We use a beam-search strategy to find the final optimum when combining the channel model and the language model.

3.2 Incrementality

Taking Johnson and Charniak's model as a starting point, we would like to develop an incremental version of that algorithm. We simulate incrementality by maintaining for each utterance to be processed an **end-of-prefix boundary**; tokens after this boundary are not available for the model to use. At each step in our incremental model, we advance this boundary by one token (the **increment**), until finally the entire utterance is available. We make use of the notion of a **prefix**, which is a substring of the utterance consisting of all tokens up to this boundary marker.

Just as in the non-incremental model, we keep track of all the possible reparanda and repairs in a chart. Every time the end-of-prefix boundary advances, we update the chart: we add all possible disfluencies which have the end position of the repair located one token before the end-of-prefix boundary, and we add all possible start points for the reparandum, interregna and repair, and end points for the reparandum and interregna, given the ordering constraints of these components.

In our basic incremental model, we leave the remainder of the algorithm untouched. When the end-of-prefix boundary reaches the end of the utterance, and thus the entire utterance is available, this model results in an identical analysis to that provided by the non-incremental model, since the chart contains identical entries, although calculated in a different order. Intuitively, this model should perform well when the current prefix is very close to being a complete utterance; and it should perform less well when a potential dis-

fluency is still under construction, since these situations are not typically found in the training data. We will return to this point further below.

We do not change the training phase of the model and we assume that the optimal values found for the non-incremental model are also optimal for the incremental model, since most weights which need to be learned are based on lexical values. Other weights are bigram based values, and values dealing with unknown tokens (i.e., tokens which occur in the test data, but not in the training data); it is not unreasonable to assume these weights are identical or very similar in both the incremental and the non-incremental model.

4 Evaluation Models and Their Application

As well as evaluating the accuracy of the analysis returned at the end of the utterance, it seems reasonable to also evaluate how quickly and accurately an incremental algorithm detects disfluencies on a word-by-word basis as the utterance is processed. In this section, we provide the methodological background to our approach, and in Section 5.2 we discuss the performance of our model when evaluated in this way.

Incremental systems are often judged solely on the basis of their output when the utterance being processed is completed. Although this does give an insight into how well a system performs overall, it does not indicate how well the incremental aspects of the mechanism perform. In this section we present an approach to the evaluation of a model of speech repair detection which measures the performance of the incremental component.

One might calculate the accuracy over all prefixes using a simple word accuracy score. However, because each prefix is a superstring of each previous prefix, such a calculation would not be fair: tokens that appear in early in the utterance will be counted more often than tokens that appear later in the utterance. In theory, the analysis of the early tokens can change at each prefix, so arguably it would make sense to reevaluate the complete analysis so far at every step. In practice, however, these changes do not happen, and so this measurement would not reflect the performance of the system correctly.

Our approach is to define a measure of **responsiveness**: that is, how soon is a disfluency detected? We propose to measure responsiveness in two ways. The **time-to-detection** score indicates how many tokens following a disfluency are read before the given disfluency is marked as one; the **delayed accuracy** score looks n tokens back from the boundary of the available utterance and, when there is a gold standard disfluency-marked token at that distance, counts how often these tokens are marked correctly.

We measure the time-to-detection score by two numbers, corresponding to the number of tokens from the start of the reparandum and the number of tokens from the start of the repair. We do this because disfluencies can be of different lengths. We assume it is unlikely that a disfluency will be found before the reparandum is completed, since the reparandum itself is often fluent. We measure the time-to-detection by the first time a given disfluency appears as one.

Since the model is a statistical model, it is possible that the most probable analysis marks a given word at position j as a disfluency, while in the next prefix the word in the same position is now no longer marked as being disfluent. A prefix later this word might be marked as disfluent again. This presents us with a problem. How do we measure when this word was correctly identified as disfluent: the first time it was marked as such or the second time? Because of the possibility of such oscillations, we take the first marking of the disfluency as the measure point. Disfluencies which are never correctly detected are not part of the time-to-detection score.

Since the evaluation starts with disfluencies found by the model, this measurement has precision-like properties only. Consequently, there are easy ways to inflate the score artificially at the cost of recall. We address this

by also calculating the delayed accuracy. This is calculated at each prefix by looking back n tokens from the prefix boundary, where $n = 0$ for the prefix boundary. For each n we calculate the accuracy score at that point over all prefixes. Each token is only assessed once given a set value of n, so we do not suffer from early prefixes being assessed more often. However, larger values of n do not take all tokens into account, since the last y tokens of an utterance will not play a part in the accuracy when $y < n$. Since we evaluate given a gold standard disfluency, this measurement has recall-like properties.

Together with the final accuracy score over the entire utterance, the time-to-detection and delayed accuracy scores provide different insights and together give a good measurement of the responsiveness and performance of the model.

Our incremental model has the same final accuracy as the original non-incremental model; this corresponds to an F-score (harmonic mean) of 0.778 on a word basis.

We found the average time to detection, measured in tokens for this model to be 8.3 measured from the start of reparandum and 5.1 from the start of repair. There are situations where disfluencies can be detected before the end of the repair; by counting from the start rather than the end of the disfluency components, we provide a way of scoring in such cases. To provide a better insight into what is happening, we also report the average distance since the start of the reparandum. We find that the time to detect is larger than the average repair length; this implies that, under this particular model, most disfluencies are only detected after the repair is finished. In fact the difference is greater than 1, which means that in most cases it takes one more token after the repair before the model identifies the disfluency.

Table 1 shows the delayed accuracy. We can see that the score first rises quickly after which the increases become much smaller. As mentioned above, a given disfluency detection in theory might oscillate. In practice, however, oscillating disfluencies are very rare, possibly because a bigram model operates on a very local level. Given that oscillation is rare, a quick stabilisation of the score indicates that, when we correctly detect a disfluency, this happens rather quickly after the disfluency has completed, since the accuracy for the large n is calculated over the same tokens as the accuracy for the smaller n (although not in the same prefix).

5 Disfluencies around Prefix Boundaries

5.1 Early detection algorithm

Our model uses a language model and a channel model to locate disfluencies. It calculates a language model probability for the utterance with the disfluency taken out, and it calculates the probability of the disfluency itself with the STAG channel model.

Consider the following example utterance fragment where a repair disfluency occurs:

$$\ldots w_i \overbrace{rn_{i+1}\ rn_{i+2}}^{\text{reparandum}}\ \overbrace{rr_{i+3}\ rr_{i+4}}^{\text{repair}}\ w_{i+5} \ldots \quad (3)$$

Here, the subscripts indicate token position in sequence; w is a token outside the disfluency; and rn is a reparandum being repaired by the repair rr. The language model estimates the continuation of the utterance without the disfluency. The model considers whether the utterance continuation after the disfluency is probable given the language model; the relevant bigram here is $p(rr_{i+3}|w_i)$, continuing with $p(rr_{i+4}|rr_{i+3})$. However, under the incremental model, it is possible the utterance has only been read as far as token $i + 3$, in which case the probability $p(w_{i+4}|w_{i+3})$ is undefined.

We would like to address the issue of looking beyond a disfluency under construction. We assume the issue of not being able to look for an utterance continuation after the repair component of the disfluency can be found back in the incremental model scores. A disfluency is usually only detected after the disfluency is completely uttered, and always requires one

n tokens back	**1**	**2**	**3**	**4**	**5**	**6**
accuracy	0.500	0.558	0.631	0.665	0.701	0.714

Table 1: delayed accuracy, n tokens back from the end of prefixes

n tokens back	**1**	**2**	**3**	**4**	**5**	**6**
accuracy	0.578	0.633	0.697	0.725	0.758	0.770

Table 2: delayed accuracy under the updated model

more token in the basic model. In the given instance this means it is unlikely that we will detect the disfluency before $i + 5$.

In order to make our model more responsive, we propose a change which makes it possible for the model to calculate channel probabilities and language model probabilities before the repair is completed. Assuming we have not yet reached the end of utterance, we would like to estimate the continuation of the utterance with the relevant bigram $p(rr_{i+4}|rr_{i+3})$. Since rr_{i+4} is not yet available we cannot calculate this probability. The correct thing to do is to sum over all possible continuations, including the end of utterance token (for the complete utterance, as opposed to the current prefix). This results in the following bigram estimation:

$$\sum_{t \in \text{vocabulary}} p(t|w_i) \tag{4}$$

This estimation is not one we need to derive from our data set, since p is a true probability. In this case, the sum over all possible continuations (this might include an end of utterance marker, in which case the utterance is already complete) equals 1. We therefore modify the algorithm so that it takes this into account. This solves the problem of the language model assessing the utterance with the disfluency cut out, when nothing from the utterance continuation after a disfluency is available.

The other issue which needs to be addressed is the alignment of the reparandum with the repair when the repair is not yet fully available. Currently the model is encouraged to align the individual tokens of the reparandum with those of the repair. The algorithm has lower estimations when the reparandum cannot be fully aligned with the repair because the reparandum and repair differ considerably in length.

We note that most disfluencies are very short: reparanda and repairs are often only one or two tokens each in length, and the interregnum is often empty. To remove the penalty for an incomplete repair, we allow the repair to grow one token beyond the prefix boundary; given the relative shortness of the disfluencies, this seems reasonable. Since this token is not available, we cannot calculate the lexical substitution value. Instead we define a new operation in the channel model: in addition to deletion, insertion, copy, and substitution, we add an additional substitution operation, the **incremental completion substitution**. This operation does not compete with the copy operation or the normal substitution operation, since it is only defined when the last token of the repair falls at the prefix boundary.

5.2 Results for the Early detection algorithm

The results of these changes are reflected in new time-to-detection and delayed accuracy scores. Again we calculated the time-to-detection, and found this to be 7.5 from the start of reparandum and 4.6 from the start of repair. Table 2 shows the results under the new early completion model using the delayed accuracy method. We see that the updated model has lower time-to-detection scores (close to a full token earlier); for delayed accuracy, we note that the scores stabilise in a similar fashion, but the scores for the updated model rise slightly more quickly.

6 Conclusions and Future Work

We have demonstrated an incremental model for finding speech disfluencies in spoken language transcripts. When we consider complete utterances, the incremental model provides identical results to those of a non-incremental model that delivers state-of-the-art accuracy in speech repair detection. We have investigated a number of measures which allow us to evaluate the model on an incremental level. Most disfluencies are identified very quickly, typically one or two tokens after the disfluency has been completed. We addressed the problems of the model around the end of prefix boundaries. These are repairs which are either still in the process of being uttered or have just been completed. We have addressed this issue by making some changes to how the model deals with prefix boundaries, and we have shown that this improves the responsiveness of the model.

The work reported in this paper uses a n-gram model as a language model and a STAG based model for the repair. We would like to replace the n-gram language model with a better language model. Previous work (Johnson and Charniak, 2004) has shown that disfluency detection can be improved by replacing the n-gram language model with a statistical parser. Besides a reported 5% accuracy improvement, this also provides a structural analysis, something which an n-gram model does not. We would like to investigate a similar extension in our incremental approach, which will require the integration of an incremental statistical parser with our noisy channel model. While transcripts of spoken texts come with manually annotated sentence boundaries, real time spoken language does not. The language model in particular takes these sentence boundaries into account. We therefore propose to investigate the properties of this model when sentence boundaries are removed.

Acknowledgements

This work was supported by the Australian Research Council as part of the Thinking Head Project, ARC/NHMRC Special Research Initiative Grant # TS0669874. We thank the anonymous reviewers for their helpful comments.

References

Charniak, Eugene. 2001. Immediate-head parsing for language models. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 124–131.

Core, Mark and Lenhart Schubert. 1999. A model of speech repairs and other disruptions. In *AAAI Fall Symposium on Psychological Models of Communication in Collaborative Systems*, pages 48–53.

Honal, Matthias and Tanja Schultz. 2003. Correction of Disfluencies in Spontaneous Speech using a Noisy-Channel Approach. In *Proceedings of the 8th Eurospeech Conference*.

Johnson, Mark and Eugene Charniak. 2004. A tag-based noisy channel model of speech repairs. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 33–39.

Marslen-Wilson, W. 1973. Linguistic structure and speech shadowing at very short latencies. *Nature*, 244:522–533.

Schuler, William, Samir AbdelRahman, Tim Miller, and Lane Schwartz. 2010. Broad-Coverage Parsing using Human-Like Memory Constraints. *Computational Linguistics*, 36(1):1–30.

Shieber, Stuart M. and Yves Schabes. 1990. Synchronous tree-adjoining grammars. In *Proceedings of the 13th International Conference on Computational Linguistics*, pages 253–258.

Shriberg, Elizabeth. 1994. *Preliminaries to a Theory of Speech Disuencies*. Ph.D. thesis, University of California, Berkeley.

Syntactic Scope Resolution in Uncertainty Analysis

Lilja Øvrelid[♠♣] and **Erik Velldal**[♣] and **Stephan Oepen**[♣]
[♣] University of Oslo, Department of Informatics
[♠]Universität Potsdam, Institut für Linguistik
ovrelid@uni-potsdam.de and erikve@ifi.uio.no and oe@ifi.uio.no

Abstract

We show how the use of syntactic structure enables the resolution of hedge scope in a hybrid, two-stage approach to uncertainty analysis. In the first stage, a Maximum Entropy classifier, combining surface-oriented and syntactic features, identifies cue words. With a small set of hand-crafted rules operating over dependency representations in stage two, we attain the best overall result (in terms of both combined ranks and average F_1) in the 2010 CoNLL Shared Task.

1 Background—Motivation

Recent years have witnessed an increased interest in the analysis of various aspects of sentiment in natural language (Pang & Lee, 2008). The subtask of *hedge resolution* deals with the analysis of uncertainty as expressed in natural language, and the linguistic means (so-called hedges) by which speculation or uncertainty are expressed. Information of this kind is of importance for various mining tasks which aim at extracting factual data. Example (1), taken from the BioScope corpus (Vincze, Szarvas, Farkas, Móra, & Csirik, 2008), shows a sentence where uncertainty is signaled by the modal verb *may*.[1]

(1) {The unknown amino acid ⟨may⟩ be used by these species}.

The topic of the Shared Task at the 2010 Conference for Natural Language Learning (CoNLL) is hedge detection in biomedical literature—in a sense 'zooming in' on one particular aspect of the broader BioNLP Shared Task in 2009 (Kim, Ohta, Pyysalo, Kano, & Tsujii, 2009). It involves two subtasks: Task 1 is described as *learning to detect sentences containing uncertainty*; the objective of Task 2 is *learning to resolve the in-sentence scope of hedge cues* (Farkas, Vincze, Mora, Csirik, & Szarvas, 2010). The organizers further suggest: *This task falls within the scope of semantic analysis of sentences exploiting syntactic patterns [...]*.

The utility of syntactic information within various approaches to sentiment analysis in natural language has been an issue of some debate (Wilson, Wiebe, & Hwa, 2006; Ng, Dasgupta, & Arifin, 2006), and the potential contribution of syntax clearly varies with the specifics of the task. Previous work in the hedging realm has largely been concerned with cue detection, i.e. identifying uncertainty cues such as *may* in (1), which are predominantly individual tokens (Medlock & Briscoe, 2007; Kilicoglu & Bergler, 2008). There has been little previous work aimed at actually resolving the scope of such hedge cues, which presumably constitutes a somewhat different and likely more difficult problem. Morante and Daelemans (2009) present a machine-learning approach to this task, using token-level, lexical information only. To this end, CoNLL 2010 enters largely uncharted territory, and it remains to be seen (a) whether syntactic analysis indeed is a necessary component in approaching this task and, more generally, (b) to what degree the specific task setup can inform us about the strong and weak points in current approaches and technology.

In this article, we investigate the contribution of syntax to hedge resolution, by reflecting on our experience in the CoNLL 2010 task.[2] Our CoNLL system submission ranked fourth (of 24) on Task 1 and third (of 15) on Task 2, for an overall best average result (there appears to be very limited overlap among top performers for the two subtasks).

[1] In examples throughout this paper, angle brackets highlight hedge cues, and curly braces indicate the scope of a given cue, as annotated in BioScope.

[2] It turns out, in fact, that all the top-performing systems in Task 2 of the CoNLLShared Task rely on syntactic information provided by parsers, either in features for machine learning or as input to manually crafted rules (Morante, Asch, & Daelemans, 2010; Rei & Briscoe, 2010).

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1379–1387,
Beijing, August 2010

	Sentences	Hedged Sentences	Cues	Multi-Word Cues	Tokens	Cue Tokens
Abstracts	11871	2101	2659	364	309634	3056
Articles	2670	519	668	84	68579	782
Total	14541	2620	3327	448	378213	3838

Table 1: Summary statistics for the Shared Task training data.

This article transcends our CoNLL system description (Velldal, Øvrelid, & Oepen, 2010) in several respects, presenting updated and improved cue detection results (§ 3 and § 4), focusing on the role of syntactic information rather than on machine learning specifics (§ 5 and § 6), providing an analysis and discussion of Task 2 errors (§ 7), and generally aiming to gauge the value of available annotated data and processing tools (§ 8). We present a hybrid, two-level approach for hedge resolution, where a statistical classifier detects cue words, and a small set of manually crafted rules operating over syntactic structures resolve scope. We show how syntactic information—produced by a data-driven dependency parser complemented with information from a 'deep', hand-crafted grammar—contributes to the resolution of in-sentence scope of hedge cues, discussing various types of syntactic constructions and associated scope detection rules in considerable detail. We furthermore present a manual error analysis, which reveals remaining challenges in our scope resolution rules as well as several relevant idiosyncrasies of the preexisting BioScope annotation.

2 Task, Data, and System Basics

Task Definition and Evaluation Metrics Task 1 is a binary sentence classification task: identifying utterances as being *certain* or *uncertain*. Following common practice, this subtask is evaluated in terms of precision, recall, and F_1 for the 'positive' class, i.e. *uncertain*. In our work, we approach Task 1 as a byproduct of the full hedge resolution problem, labeling a sentence as *uncertain* if it contains at least one token classified as a hedge cue. In addition to the sentence-level evaluation for Task 1, we also present precision, recall, and F_1 for the cue-level.

Task 2 comprises two subtasks: cue detection and scope resolution. The official CoNLL evaluation does not tease apart these two aspects of the problem, however: Only an exact match of both the cue and scope bracketing (in terms of substring positions) will be counted as a success, again quantified in terms of precision, recall, and F_1. Discussing our results below, we report cue detection and scope resolution performance separately, and further put scope results into perspective against an upper bound based on the gold-standard cue annotation.

Besides the primary biomedical domain data, some annotated Wikipedia data was provided for Task 1, and participating systems are classified as *in-domain* (using exclusively the domain-specific data), *cross-domain* (combining both types of training data), or *open* (utilizing additional uncertainty-related resources). In our work, we focus on the interplay of syntax and the more challenging Task 2; we ignored the Wikipedia track in Task 1. Despite our using general NLP tools (see below), our system falls into the most restrictive, *in-domain* category.

Training and Evaluation Data The training data for the CoNLL 2010 Shared Task is taken from the BioScope corpus (Vincze et al., 2008) and consists of 14,541 'sentences' (or other root-level utterances) from biomedical abstracts and articles (see Table 1).[3] The BioScope corpus provides annotation for hedge cues as well as their scope. According to the annotation guidelines (Vincze et al., 2008), the annotation adheres to a principle of minimalism when it comes to hedge cues, i.e. the minimal unit expressing hedging is annotated. The inverse is true of scope annotations, which adhere to a principle of maximal scope—meaning that scope should be set to the largest syntactic

[3] As it was known beforehand that evaluation would draw on full articles only, we put more emphasis on the article subset of the training data, for example in cross validation testing and manual diagnosis of errors.

ID	FORM	LEMMA	POS	FEATS	HEAD	DEPREL	XHEAD	XDEP
1	The	the	DT	_	4	NMOD	4	SPECDET
2	unknown	unknown	JJ	degree:attributive	4	NMOD	4	ADJUNCT
3	amino	amino	JJ	degree:attributive	4	NMOD	4	ADJUNCT
4	acid	acid	NN	pers:3|case:nom|num:sg|ntype:common	5	SBJ	3	SUBJ
5	may	may	MD	mood:ind|subcat:MODAL|tense:pres|clauseType:decl	0	ROOT	0	ROOT
6	be	be	VB	_	5	VC	7	PHI
7	used	use	VBN	subcat:V-SUBJ-OBJ|vtype:main|passive:+	6	VC	5	XCOMP
8	by	by	IN	_	7	LGS	9	PHI
9	these	these	DT	deixis:proximal	10	NMOD	10	SPECDET
10	species	specie	NNS	num:pl|pers:3|case:obl|common:count|ntype:common	8	PMOD	7	OBL-AG
11	.	.	.	_	5	P	0	PUNC

Table 2: Stacked dependency representation of example (1), with MaltParser and XLE annotations.

unit possible.

For evaluation purposes, the task organizers provided newly annotated biomedical articles, following the same general BioScope principles. The CoNLL 2010 evaluation data comprises 5,003 additional utterances (138,276 tokens), of which 790 are annotated as hedged. The data contains a total of 1033 cues, of which 87 are so-called multiword cues (i.e. cues spanning multiple tokens), comprising 1148 cue tokens altogether.

Stacked Dependency Parsing For syntactic analysis we employ the open-source MaltParser (Nivre, Hall, & Nilsson, 2006), a platform for data-driven dependency parsing. For improved accuracy and portability across domains and genres, we make our parser incorporate the predictions of a large-scale, general-purpose LFG parser—following the work of Øvrelid, Kuhn, and Spreyer (2009). A technique dubbed *parser stacking* enables the data-driven parser to learn, not only from gold standard treebank annotations, but from the output of another parser (Nivre & McDonald, 2008). This technique has been shown to provide significant improvements in accuracy for both English and German (Øvrelid et al., 2009), and a similar setup employing an HPSG grammar has been shown to increase domain independence in data-driven dependency parsing (Zhang & Wang, 2009). The stacked parser combines two quite different approaches—data-driven dependency parsing and 'deep' parsing with a handcrafted grammar—and thus provides us with a broad range of different types of linguistic information for the hedge resolution task.

MaltParser is based on a deterministic parsing strategy in combination with treebank-induced classifiers for predicting parse transitions. It supports a rich feature representation of the parse history in order to guide parsing and may easily be extended to take additional features into account. The procedure to enable the data-driven parser to learn from the grammar-driven parser is quite simple. We parse a treebank with the XLE platform (Crouch et al., 2008) and the English grammar developed within the ParGram project (Butt, Dyvik, King, Masuichi, & Rohrer, 2002). We then convert the LFG output to dependency structures, so that we have two parallel versions of the treebank—one gold standard and one with LFG annotation. We extend the gold standard treebank with additional information from the corresponding LFG analysis and train MaltParser on the enhanced data set.

Table 2 shows the enhanced dependency representation of example (1) above, taken from the training data. For each token, the parsed data contains information on the word form, lemma, and part of speech (PoS), as well as on the head and dependency relation in columns 6 and 7. The added XLE information resides in the FEATS column, and in the XLE-specific head and dependency columns 8 and 9. Parser outputs, which in turn form the basis for our scope resolution rules discussed in Section 5, also take this same form. The parser employed in this work is trained on the Wall Street Journal sections 2 – 24 of the Penn Treebank (PTB), converted to dependency format (Johansson & Nugues, 2007) and extended with XLE features, as described above. Parsing uses the arc-eager mode of MaltParser and an SVM with a polynomial kernel. When tested using 10-fold cross validation on the enhanced PTB, the parser achieves a labeled accuracy score of 89.8.

PoS Tagging and Domain Variation Our parser is trained on financial news, and although stacking with a general-purpose LFG parser is ex-

pected to aid domain portability, substantial differences in domain and genre are bound to negatively affect syntactic analysis (Gildea, 2001). MaltParser presupposes that inputs have been PoS tagged, leaving room for variation in preprocessing. On the one hand, we aim to make parser inputs maximally similar to its training data (i.e. the conventions established in the PTB); on the other hand we wish to benefit from specialized resources for the biomedical domain.

The GENIA tagger (Tsuruoka et al., 2005) is particularly relevant in this respect (as could be the GENIA Treebank proper[4]). However, we found that GENIA tokenization does not match the PTB conventions in about one out of five sentences (for example wrongly splitting tokens like '`390,926`' or '`Ca(2+)`'); also in tagging proper nouns, GENIA systematically deviates from the PTB. Hence, we adapted an in-house tokenizer (using cascaded finite-state rules) to the CoNLL task, run two PoS taggers in parallel, and eclectically combine annotations across the various preprocessing components—predominantly giving precedence to GENIA lemmatization and PoS hypotheses.

To assess the impact of improved, domain-adapted inputs on our hedge resolution system, we contrast two configurations: first, running the parser in the exact same manner as Øvrelid, Kuhn, and Spreyer (2010), we use TreeTagger (Schmid, 1994) and its standard model for English (trained on the PTB) for preprocessing; second, we give as inputs to the parser our refined tokenization and merged PoS tags, as described above. When evaluating the two modes of preprocessing on the articles subset of the training data, and using gold-standard cues, our system for resolving cue scopes (presented in § 5) achieves an F_1 of 66.31 with TreeTagger inputs, and 72.30 using our refined tokenization and tagger combination. These results underline the importance of domain adaptation for accurate syntactic analysis, and in the following we assume our hybrid in-house setup.

[4]Although the GENIA Treebank provides syntactic annotation in a form inspired by the PTB, it does not provide function labels. Therefore, our procedure for converting from constituency to dependency requires non-trivial adaptation before we can investigate the effects of retraining the parser against GENIA.

3 Stage 1: Identifying Hedge Cues

For the task of identifying hedge cues, we developed a binary maximum entropy (MaxEnt) classifier. The identification of cue words is used for (a) classifying sentences as certain/uncertain (Task 1), and (b) providing input to the syntactic rules that we later apply for resolving the in-sentence scope of the cues (Task 2). We also report evaluation scores for the sub-task of cue detection in isolation.

As annotated in the training data, it is possible for a hedge cue to span multiple tokens, e.g. as in *whether or not*. The majority of the multi-word cues in the training data are very infrequent, however, most occurring only once, and the classifier itself is not sensitive to the notion of multi-word cues. Instead, the task of determining whether a cue word forms part of a larger multi-word cue, is performed in a separate post-processing step (applying a heuristic rule targeted at only the most frequently occurring patterns of multi-word cues in the training data).

During development, we trained cue classifiers using a wide variety of feature types, both syntactic and surface-oriented. In the end, however, we found n-gram-based lexical features to have the greatest contribution to classifier performance. Our best-performing classifier so far (see 'Final' in Table 3) includes the following feature types: n-grams over forms (up to 2 tokens to the right), n-grams over base forms (up to 3 tokens left and right), PoS (from GENIA), subcategorization frames (from XLE), and phrase-structural coordination level (from XLE). Our CoNLL system description includes more details of the various other feature types that we experimented with (Velldal et al., 2010).

4 Cue Detection Evaluation

Table 3 summarizes the performance of our MaxEnt hedge cue classifier in terms of precision, recall and F_1, computed using the official Shared Task scorer script. The sentence-level scores correspond to Task 1 of the Shared Task, and the cue-level scores are based on the exact-match counts for full hedge cues (possibly spanning multiple tokens).

Configuration	Sentence Level			Cue Level		
	Prec	Rec	F_1	Prec	Rec	F_1
Baseline, Development	79.25	79.45	79.20	77.37	71.70	74.43
Final, Development	91.39	86.78	89.00	90.18	79.47	84.49
Final, Held-Out	85.61	85.06	85.33	81.97	76.41	79.10

Table 3: Isolated evaluation of the hedge cue classifier.

As the CoNLL test data was known beforehand to consist of articles only, in 10-fold cross validation for classifier development we tested exclusively against the articles segment, while always including all sentences from the abstracts in the training set. This corresponds to the development results in Table 3, while the held-out results are for the official Shared Task evaluation data (training on all the available training data). A model using only unigram features serves as a baseline.

5 Stage 2: Resolving Scope

Hedge scope may vary quite a lot depending on linguistic properties of the cue in question. In our approach to scope resolution we rely heavily on syntactic information, taken from the dependency structures proposed by both MaltParser and XLE, as well as on various additional features relating to specific syntactic constructions.

We constructed a small set of heuristic rules which define the scope for each cue detected in Stage 1. In developing these rules, we made use of the information provided by the guidelines for scope annotation in the BioScope corpus (Vincze et al., 2008), combined with manual inspection of the training data in order to further generalize over the phenomena discussed by Vincze et al. (2008) and work out interactions of constructions for various types of cues.

The rules take as input a parsed sentence which has been further tagged with hedge cues. They operate over the dependency structures and additional features provided by the parser. Default scope is set to start at the cue word and span to the end of the sentence (modulo punctuation), and this scope also provides the baseline for the evaluation of our rules. In the following, we discuss broad classes of rules, organized by categories of hedge cues. As there is no explicit representation of phrase or clause boundaries in our dependency universe, we assume a set of functions over dependency graphs, for example finding the left- or rightmost (direct) *dependent* of a given node, or transitively selecting left- or rightmost *descendants*.

Coordination The dependency analysis of coordination provided by our parser makes the first conjunct the head of the coordination. For cues that are coordinating conjunctions (PoS tag CC), such as *or*, we define the scope as spanning the whole coordinate structure, i.e. start scope is set to the leftmost dependent of the head of the coordination, e.g., *roX* in (2), and end scope is set to its rightmost dependent (conjunct), e.g., *RNAs* in (2). This analysis provides us with coordinations at various syntactic levels, such as NP and $\overline{\text{N}}$ (2), AP and AdvP, or VP (3):

(2) [...] the {roX genes ⟨or⟩ RNAs} recruit the entire set of MSL proteins [...]

(3) [...] the binding interfaces are more often {kept ⟨or⟩ even reused} rather than lost in the course of evolution.

Adjectives We distinguish between adjectives (JJ) in *attributive* (NMOD) function and adjectives in *predicative* (PRD) function. Attributive adjectives take scope over their (nominal) head, with all its dependents, as in (4) and (5):

(4) The {⟨possible⟩ selenocysteine residues} are shown in red, [...]

(5) Extensive analysis of the flanks failed to show any hallmarks of {⟨putative⟩ transposons that might be associated with this RAG1-like protein}, [...]

For adjectives in a predicative function the scope includes the subject argument of the head verb (the copula), as well as a (possible) clausal argument, as in (6). The scope does not, however, include expletive subjects, as in (7).

(6) Therefore, {the unknown amino acid, if it is encoded by a stop codon, is ⟨unlikely⟩ to exist in the current databases of microbial genomes}.

(7) For example, it is quite {⟨likely⟩ that there exists an extremely long sequence that is entirely unique to U}.

Verbs The scope of verbal cues is a bit more complex and depends on several factors. In our rules, we distinguish *passive* usages from active usages, *raising* verbs from non-raising verbs, and the presence or absence of a subject-control embedding context. The scopes of both passive and raising verbs include the subject argument of their head verb, as in (8) and (9), unless it is an expletive pronoun, as in (10).

(8) {Interactions determined by high-throughput methods are generally ⟨considered⟩ to be less reliable than those obtained by low-throughput studies} 1314 and as a consequence [...]

(9) {Genomes of plants and vertebrates ⟨seem⟩ to be free of any recognizable Transib transposons} (Figure 1).

(10) It has been {⟨suggested⟩ that unstructured regions of proteins are often involved in binding interactions, particularly in the case of transient interactions} 77.

In the case of subject control involving a hedge cue, specifically modals, subject arguments are included in scopes where the controller heads a passive construction or a raising verb, as in example (1) above, repeated here for convenience:

(11) {The unknown amino acid ⟨may⟩ be used by these species}.

In general, the end scope of verbs should extend over the minimal clause that contains the verb in question. In terms of dependency structures, we define the clause boundary as comprising the chain of descendants of a verb which is not intervened by a token with a higher attachment in the graph than the verb in question. In example (8) for instance, the sentence-level conjunction *and* marks the end of the clause following the cue *considered*.

Prepositions and Adverbs Cues that are tagged as prepositions (including some complementizers) take scope over their argument, with all its descendants, (12). Adverbs take scope over their head with all its (non-subject) syntactic descendants (13).

	Configuration	F_1
BSP	Default, Gold Cues	45.21
	Rules, Gold Cues	72.31
	Rules, System Cues	64.77
BSE	Rules, Gold Cues	66.73
	Rules, System Cues	55.75

Table 4: Evaluation of scope resolution rules.

(12) {⟨Whether⟩ the codon aligned to the inframe stop codon is a nonsense codon or not} was neglected at this stage.

(13) These effects are {⟨probably⟩ mediated through the 1,25(OH)2D3 receptor}.

Multi-Word Cues In the case of multi-word cues, such as *indicate that* or *either ... or*, we need to determine the head of the multi-word unit. We then set the scope of the whole unit to the scope of the head token.

As an illustration of rule processing, consider our running example (11), with its syntactic analysis as shown in Table 2 above. This example invokes a variety of syntactic properties, including parts of speech, argumenthood, voice etc. Initially, the scope of the hedge cue is set to default scope. Then the subject control rule is applied, which checks the properties of the verbal argument *used*, going through a chain of verbal dependents from the modal verb. Since it is marked as passive in the LFG analysis, the start scope is set to include the subject of the cue word (the leftmost descendant in its SBJ dependent).

6 Rule Evaluation

Table 4 summarizes scope resolution performance (viewed as a an isolated subtask) for various configurations, both against the articles section of the CoNLL training data (dubbed BSP) and against the held-out evaluation data (BSE). First of all, we note that the 'default scope' baseline is quite strong: unconditionally extending the scope of a cue to the end of the sentence yields an F_1 of 45.21. Given gold standard cue information, our scope rules improve on the baseline by 27 points on the articles section of the data set, for an F_1 of 72.31; with system-assigned hedge cues, our rules still

achieve an F_1 of 64.77. Note that scope resolution scores based on classified cues also yield the end-to-end system evaluation for Task 2.

The bottom rows of Table 4 show the evaluation of scope rules on the CoNLL held-out test data. Using system cues, scope resolution on the held-out data scores at 55.75 F_1. Comparing to the result on the (articles portion of the) training data, we observe a substantial drop in performance (of six points with gold-standard cues, nine points with system cues). There are several possible explanations for this effect. First of all, there may well be a certain degree of overfitting of our rules to the training data. The held-out data may contain hedging constructions that are not covered by our current set of scope rules, or annotation of parallel constructions may in some cases differ in subtle ways (see § 7 below). Moreover, scope resolution performance is of course influenced by cue detection (see Table 3). The cue-level F_1 of our system on the held-out data set is 79.10, compared to 84.49 (using cross validation) on the training data. This drop in cue-level performance appears to affect classification precision far more than recall. Of course, given that our heuristics for identifying multi-word cues were based on patterns extracted from the training data, some loss in the cue-level score was expected.

7 Error Analysis

To start shedding some light on the significance of our results, we performed a manual error analysis on the article portion of the training material (BSP), with two of the authors (trained linguists) working in tandem. Using gold-standard cues, our scope resolution rules fail to exactly replicate the target annotation in 185 (of 668) cases, corresponding to 72.31 F_1 in Table 4 above. Our evaluators reviewed and discussed these 185 cases, classifying 156 (84 %) as genuine system errors, 22 (12 %) as likely[5] annotation errors, and a remaining seven cases as involving controversial or seemingly arbitrary decisions.

The two most frequent classes of system errors pertain (a) to the recognition of phrase and clause boundaries and (b) to not dealing successfully with relatively superficial properties of the text. Examples (14) and (15) illustrate the first class of errors, where in addition to the gold-standard annotation we use vertical bars ('|') to indicate scope predictions of our system.

(14) [...] {the reverse complement |mR of m will be ⟨considered⟩ to be [...]|}

(15) This |{⟨might⟩ affect the results} if there is a systematic bias on the composition of a protein interaction set|.

In our syntax-driven approach to scope resolution, system errors will almost always correspond to a failure in determining constituent boundaries, in a very general sense. However, specifically example (15) is indicative of a key challenge in this task, where adverbials of condition, reason, or contrast frequently attach within the dependency domain of a hedge cue, yet are rarely included in the scope annotation.

Example (16) demonstrates our second frequent class of system errors. One in six items in the BSP training data contains a sentence-final parenthesized element or trailing number, as for example (2), (9), or (10) above; most of these are bibliographic or other in-text references, which are never included in scope annotation. Hence, our system includes a rule to 'back out' from trailing parentheticals; in examples like (16), however, syntax does not make explicit the contrast between an in-text reference vs. another type of parenthetical.

(16) More specifically, {|the bristle and leg phenotypes are ⟨likely⟩ to result from reduced signaling by Dl| (and not by Ser)}.

Moving on to apparent annotation errors, the rules for inclusion (or not) of the subject in the scope of verbal hedge cues and decisions on boundaries (or internal structure) of nominals

[5] In some cases, there is no doubt that annotation is erroneous, i.e. in violation of the available annotation guidelines (Vincze et al., 2008) or in conflict with otherwise unambiguous patterns. In other cases, however, judgments are necessarily based on generalizations made by the evaluators, i.e. assumptions about the underlying system and syntactic analyses implicit in the BioScope annotations. Furthermore, selecting items for manual analysis that do not align with the predictions made by our scope resolution rules is likely to bias our sample, such that our estimated proportion of 12 % annotation errors cannot be used to project an overall error rate.

seem problematic—as illustrated in examples (17) to (22).[6]

(17) [...] and |this is also {⟨thought⟩ to be true for the full protein interaction networks we are modeling}|.

(18) [...] {Neur |⟨can⟩ promote Ser signaling|}.

(19) |Some of the domain pairs {⟨seem⟩ to mediate a large number of protein interactions, thus acting as reusable connectors}|.

(20) One {|⟨possible⟩ explanation| is functional redundancy with the mouse Neur2 gene}.

(21) [...] |redefinition of {one of them is ⟨feasible⟩}|.

(22) |The {Bcl-2 family ⟨appears⟩ to function [...]}|.

Finally, the difficult corner cases invoke non-constituent coordination, ellipsis, or NP-initial focus adverbs—and of course interactions of the phenomena discussed above. Without making the syntactic structures assumed explicit, it is often very difficult to judge such items.

8 Reflections — Outlook

Our combination of stacked dependency parsing and hand-crafted scope resolution rules proved adequate for the CoNLL 2010 competition, confirming the central role of syntax in this task. With a comparatively small set of rules (implemented in a few hundred lines of code), constructed through roughly two full weeks of effort (studying BioScope annotations and developing rules), our CoNLL system achieved an end-to-end F_1 of 55.33 on Task 2.[7] The two submissions with better results (at 57.32 and 55.65 F_1) represent groups who have pioneered the hedge analysis task in previous years (Morante et al., 2010; Rei & Briscoe, 2010). Scores for other 'in-domain' participants range from 52.24 to 2.15 F_1. Doubtless there is room for straightforward extension: for example retraining our parser on the GENIA Treebank, further improving the cue classifier, and refining scope resolution rules in the light of the error analysis above.

At the same time, we remain mildly ambivalent about the long-term impact of some of the specifics of the 2010 CoNLL task. Shared tasks (i.e. system bake-offs) have become increasingly popular in past years, and in some subfields (e.g. IE, SMT, or dependency parsing) high-visibility competitions can shape community research agendas. Hence, even at this early stage, it seems appropriate to reflect on the possible conclusions to be drawn from the 2010 hedge resolution task. First, we believe the harsh 'exact substring match' evaluation metric underestimates the degree to which current technology can solve this problem; furthermore, idiosyncratic, string-level properties (e.g. the exact treatment of punctuation or parentheticals) may partly obscure the interpretation of methods used and corresponding system performance.

These effects are compounded by some concerns about the quality of available annotation. Even though we tried fine-tuning our cross validation testing to the nature of the evaluation data (comprising only articles), our system performs substantially worse on the newly annotated CoNLL test data, in both stages.[8] In our view, the annotation of hedge cues and scopes ideally would be overtly related to at least some level of syntactic annotation—as would in principle be possible for the segment of BioScope drawing on the abstracts of the GENIA Treebank.

Acknowledgements

We are grateful to the organizers of the 2010 CoNLL Shared Task and creators of the BioScope resource; first, for engaging in these kinds of community service, and second for many in-depth discussions of annotation and task details. We thank our colleagues at the Universities of Oslo and Potsdam for their comments and support.

[6] Like in the presentation of system errors, we include scope predictions of our own rules here too, which we believe to be correct in these cases. Also in this class of errors, we find the occasional 'uninteresting' mismatch, for example related to punctuation marks and inconsistencies around parentheses.

[7] In § 4 and § 6 above, we report scores for a slightly improved version of our system, where (after the official CoNLL submission date) we eliminated a bug related to the treatment of sentence-initial whitespace in the XML annotations. At an end-to-end F_1 of 55.75, this system would outrank the second best performer in Task 2.

[8] We are leaving open the possibility to further refine our system; we have therefore abstained from an error analysis on the evaluation data so far.

References

Butt, M., Dyvik, H., King, T. H., Masuichi, H., & Rohrer, C. (2002). The Parallel Grammar Project. In *Proceedings of COLING workshop on grammar engineering and evaluation* (pp. 1 – 7). Taipei, Taiwan.

Crouch, D., Dalrymple, M., Kaplan, R., King, T., Maxwell, J., & Newman, P. (2008). *XLE documentation.* Palo Alto, CA. (Palo Alto Research Center)

Farkas, R., Vincze, V., Mora, G., Csirik, J., & Szarvas, G. (2010). The CoNLL 2010 Shared Task: Learning to detect hedges and their scope in natural language text. In *Proceedings of the 14th Conference on Natural Language Learning.* Uppsala, Sweden.

Gildea, D. (2001). Corpus variation and parser performance. In *Proceedings of the 2001 conference on Empirical Methods in Natural Language Processing* (pp. 167 – 202). Pittsburgh, PA.

Johansson, R., & Nugues, P. (2007). Extended constituent-to-dependency conversion for English. In J. Nivre, H.-J. Kaalep, & M. Koit (Eds.), *Proceedings of NODALIDA 2007* (p. 105-112). Tartu, Estonia.

Kilicoglu, H., & Bergler, S. (2008). Recognizing speculative language in biomedical research articles: A linguistically motivated perspective. In *Proceedings of the BioNLP 2008 Workshop.* Columbus, OH, USA.

Kim, J.-D., Ohta, T., Pyysalo, S., Kano, Y., & Tsujii, J. (2009). Overview of BioNLP 2009 Shared Task on event extraction. In *Proceedings of the BioNLP 2009 workshop companion volume for shared task* (pp. 1 – 9). Boulder, CO: Association for Computational Linguistics.

Medlock, B., & Briscoe, T. (2007). Weakly supervised learning for hedge classification in scientific literature. In *Proceedings of the 45th Meeting of the Association for Computational Linguistics* (pp. 992 – 999). Prague, Czech Republic: Association for Computational Linguistics.

Morante, R., Asch, V. V., & Daelemans, W. (2010). Memory-based resolution of in-sentence scope of hedge cues. In *Proceedings of the 14th Conference on Natural Language Learning* (pp. 40 – 47). Uppsala, Sweden.

Morante, R., & Daelemans, W. (2009). Learning the scope of hedge cues in biomedical texts. In *Proceedings of the BioNLP 2009 Workshop* (pp. 28 – 36). Boulder, Colorado.

Ng, V., Dasgupta, S., & Arifin, S. M. N. (2006). Examining the role of linguistic knowledge sources in the automatic identification and classification of reviews. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics.* Sydney, Australia.

Nivre, J., Hall, J., & Nilsson, J. (2006). MaltParser: A data-driven parser-generator for dependency parsing. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation* (p. 2216-2219). Genoa, Italy.

Nivre, J., & McDonald, R. (2008, June). Integrating graph-based and transition-based dependency parsers. In *Proceedings of the 46th Meeting of the Association for Computational Linguistics* (pp. 950 – 958). Columbus, Ohio.

Øvrelid, L., Kuhn, J., & Spreyer, K. (2009). Improving data-driven dependency parsing using large-scale LFG grammars. In *Proceedings of the 47th Meeting of the Association for Computational Linguistics* (pp. 37 – 40). Singapore.

Øvrelid, L., Kuhn, J., & Spreyer, K. (2010). Cross-framework parser stacking for data-driven dependency parsing. *TAL 2010 special issue on Machine Learning for NLP*, *50*(3).

Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, *2*(1-2).

Rei, M., & Briscoe, T. (2010). Combining manual rules and supervised learning for hedge cue and scope detection. In *Proceedings of the 14th Conference on Natural Language Learning* (pp. 56 – 63). Uppsala, Sweden.

Schmid, H. (1994). Probabilistic part-of-speech tagging using decision trees. In *International conference on new methods in language processing* (p. 44-49). Manchester, England.

Tsuruoka, Y., Tateishi, Y., Kim, J.-D., Ohta, T., McNaught, J., Ananiadou, S., et al. (2005). Developing a robust Part-of-Speech tagger for biomedical text. In *Advances in informatics* (pp. 382 – 392). Berlin, Germany: Springer.

Velldal, E., Øvrelid, L., & Oepen, S. (2010). Resolving speculation: MaxEnt cue classification and dependency-based scope rules. In *Proceedings of the 14th Conference on Natural Language Learning.* Uppsala, Sweden.

Vincze, V., Szarvas, G., Farkas, R., Móra, G., & Csirik, J. (2008). The BioScope corpus: Annotation for negation, uncertainty and their scope in biomedical texts. In *Proceedings of the BioNLP 2008 Workshop.* Columbus, OH, USA.

Wilson, T., Wiebe, J., & Hwa, R. (2006). Recognizing strong and weak opinion clauses. *Computational Intelligence*, *22*(2), 73 – 99.

Zhang, Y., & Wang, R. (2009). Cross-domain dependency parsing using a deep linguistic grammar. In *Proceedings of the 47th Meeting of the Association for Computational Linguistics.* Singapore.