Semi-supervised Semantic Pattern Discovery with Guidance from Unsupervised Pattern Clusters

Ang Sun
Computer Science Department
New York University
asun@cs.nyu.edu

Ralph Grishman
Computer Science Department
New York University
grishman@cs.nyu.edu

Abstract

We present a simple algorithm for clustering semantic patterns based on distributional similarity and use cluster memberships to guide semi-supervised pattern discovery. We apply this approach to the task of relation extraction. The evaluation results demonstrate that our novel bootstrapping procedure significantly outperforms a standard bootstrapping. Most importantly, our algorithm can effectively prevent semantic drift and provide semi-supervised learning with a natural stopping criterion.

1 Introduction

The Natural Language Processing (NLP) community faces new tasks and new domains all the time. Without enough labeled data of a new task or a new domain to conduct supervised learning, semi-supervised learning (SSL) is particularly attractive to NLP researchers since it only requires a handful of labeled examples, known as seeds. SSL starts with these seeds to train an initial model; it then applies this model to a large volume of unlabeled data to get more labeled examples and adds the most confident ones as new seeds to re-train the model. This iterative procedure has been successfully applied to a variety of NLP tasks, such as hypernym/hyponym extraction (Hearst, 1992), word sense disambiguation (Yarowsky, 1995), question answering (Ravichandran and Hovy, 2002), and information extraction (Brin, 1998; Collins and Singer, 1999; Riloff and Jones, 1999; Agichtein and Gravano, 2000; Yangarber et al., 2000; Chen and Ji, 2009).

While SSL can give good performance for many tasks, it is a procedure born with two defects. One is semantic drift. When SSL is under-constrained, the semantics of newly promoted examples might stray away from the original meaning of seed examples as discussed in (Brin, 1998; Curran et al., 2007; Carlson et al., 2010). For example, a SSL procedure to learn semantic patterns for the *LocatedIn* relation (PERSON in LOCATION/GPE[1]) might accept patterns for the *Employment* relation (employee of GPE / ORGANIZATION) because many unlabeled pairs of names are connected by patterns belonging to multiple relations. Patterns connecting <*Bill Clinton, Arkansas*> include *LocatedIn* patterns such as "*visit*", "*arrive in*" and "*fly to*", but also patterns indicating other relations such as "*governor of*", "*born in*", and "*campaign in*". Similar analyses can be applied to many other examples such as <*Bush, Texas*> and <*Schwarzenegger, California*>. Without careful design, SSL procedures usually accept bogus examples during certain iterations and hence the learning quality degrades.

The other shortcoming of SSL is its lack of natural stopping criteria. Most SSL algorithms either run a fixed number of iterations (Agichtein and Gravano, 2000) or run against a separate labeled test set to find the best stopping criterion (Abney, 2008). The former solution needs a human to keep eyeballing the learning quality of different iterations and set ad-hoc thresholds accordingly. The latter requires a

[1] These are the types of relations and names used in the NIST-sponsored ACE evaluation. http://www.itl.nist.gov/iad/mig//tests/ace/. GPE represents a Geo-Political Entity — an entity with land and a government.

Coling 2008: Poster Volume, pages 1194–1202,
Beijing, August 2010

separate labeled test set for each new task or domain. They make SSL less appealing than it could be since the intention of using SSL is to minimize supervision.

In this paper, we propose a novel learning framework which can automatically monitor the semantic drift and find a natural stopping criterion for SSL. Central to our idea is that instead of using unlabeled data directly in SSL, we first cluster the seeds and unlabeled data in an unsupervised way before conducting SSL. The semantics of unsupervised clusters are usually unknown. However, the cluster to which the seeds belong can serve as the target cluster. Then we guide the SSL procedure using the target cluster. Under such learning settings, semantic drift can be automatically detected and a stopping criterion can be found: stopping the SSL procedure when it tends to accept examples belonging to clusters other than the target cluster.

We demonstrate in this paper the above general idea by considering a bootstrapping procedure to discover semantic patterns for extracting relations between named entities (NE). Standard bootstrapping usually starts with some high-precision and high frequency seed patterns for a specific relation to match named entities, then it uses newly promoted entities to search for additional confident patterns connecting them. It is a procedure driven by the duality between patterns and entities: a good pattern can connect more than one pair of named entities and a pair of named entities is usually connected by more than one good pattern.

We present a new bootstrapping procedure in which we first cluster the seed and other patterns in a large corpus based on distributional similarity. We then guide the bootstrapping using the target cluster.

The next section describes our unsupervised pattern clusters. Section 3 presents the details of our novel bootstrapping procedure with guidance from pattern clusters. We evaluate our algorithms in Section 4 and present related work in Section 5. We draw conclusions and point to future work in Section 6.

2 Pattern Clusters

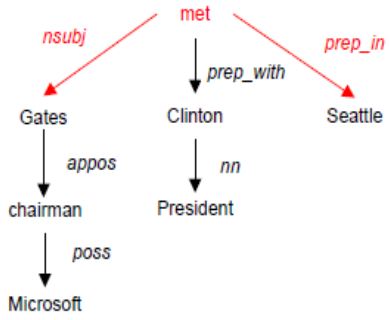
2.1 Distributional Hypothesis

The Distributional Hypothesis (Harris, 1954) states that words that tend to occur in similar contexts tend to have similar meanings. Lin and Pantel (2001) extended this hypothesis to cover patterns (dependency paths in their case). The idea of the extension is that if two patterns tend to occur in similar contexts then the meanings of the patterns tend to be similar. For example, in "*X solves Y*" and "*X finds a solution to Y*", "*solves*" and "*finds a solution to*" share many common *Xs* and *Ys* and hence are similar to each other. This extended distributional hypothesis serves as the basis on which we compute similarities for each pair of patterns.

2.2 Pattern Representation — Shortest Dependency Path

We adopt a shortest dependency path (SDP) representation of relation patterns. SDP has demonstrated its power in kernel methods for relation extraction (Bunescu and Mooney, 2005). Its capability in capturing most of the information of interest is also evidenced by a systematic comparison of effectiveness of different information extraction (IE) patterns in (Stevenson and Greenwood, 2006) [2] . For example, "*nsubj* ← met → *prep_in*" is able to represent *LocatedIn* between "*Gates*" and "*Seattle*" while a token-based pattern would be much less general because it would have to specify all the intervening tokens.

Figure 1. Stanford dependency tree for sentence *"Gates, Microsoft's chairman, met with President Clinton in Seattle".*



[2] SDP is equivalent to the linked chains described in Stevenson and Greenwood (2006) when the dependency of a sentence is represented as a tree not a graph.

2.3 Pre-processing

We tag and parse each sentence in our corpus with the NYU named entity tagger[3] and the Stanford dependency parser. Then for each pair of names in the dependency tree, we extract the SDP connecting them. Names in the path are replaced by their types. We require SDP to contain at least one verb or noun. We use the base form of words in SDP. We also require the length of the path (defined as the number of dependency relations and words in it) to be between 3 and 7. Short paths are more likely to be generic patterns such as "*of*" and can be handled separately as in (Pantel and Pennacchiotti, 2006). Very long paths are more likely to be non-relation patterns and too sparse to be useful even if they are relation patterns.

2.4 Clustering Algorithm

The basic idea of our clustering algorithm is to group all the paths (including the seed paths used later for SSL) in our corpus into different clusters based on distributional similarities. We first extract a variety of features from the named entities *X* and *Y* connected by a path *P* as shown in Table 1. We then compute an analogue of tf-idf for each feature *f* of *P* as follows: tf as the number of corpus instances of *P* having feature *f* divided by the number of instances of *P*; idf as the total number of paths in the corpus divided by the number of paths with at least one instance with feature *f*. Then we adopt a vector space model, i.e., we construct a tf-idf feature vector for each *P*. Now we compute the similarity between two vectors/paths using Cosine similarity and cluster all the paths using Complete Linkage.

Some technical details deserve more attention here.

Feature extraction: We extract more types of features than the DIRT paraphrase discovery procedure used in (Lin and Pantel, 2001). Lin and Pantel (2001) considered *X* and *Y* separately while we also use the conjunction of *X* and *Y*. We also extract named entity types as features since we are interested in discovering relations among different types of names. Some names are ambiguous such as *Jordan*. We hope coupling the type with the string of the name may alleviate the ambiguity.

Table 1. Sample features for "*X visited Y*" as in "*Jordan visited China*"

Feature Type	Example
Name Type of *X*	*LEFT_PERSON*
Name Type of *Y*	*RIGHT_GPE*
Combination of Types of X and *Y*	*PERSON_GPE*
Conjunction of String and Type of *X*	*LEFT_Jordan_PERSON*
Conjunction of String and Type of *Y*	*RIGHT_China_GPE*
Conjunction of Strings and Types of *X* and *Y*	*Jordan_PERSON_China_GPE*

Similarity measure and clustering method: There are many ways to compute the similarity/distance between two feature vectors, such as *Cosine*, *Euclidean*, *Hamming*, and *Jaccard coefficient*. There are also many standard clustering algorithms. A systematic comparison of the performance of different distance measures and clustering algorithms is beyond the scope of this paper.

3 Semi-supervised Relation Pattern Discovery

We first present a standard bootstrapping algorithm coupled with analyses of some of its shortcomings. Then we describe our new bootstrapping procedure which is guided by pattern clusters.

3.1 Bootstrapping without Guidance

The procedure associates a precision between 0 and 1 with each pattern, and a confidence between 0 and 1 with each name pair. Initially the seed patterns for a specific relation *R* have precision 1 and all other patterns 0. It consists of the following steps:

Step1: Use seed patterns to match new NE pairs and evaluate NE pairs.

Intuitively, for a newly matched NE pair N_i, if many of the *k* patterns connecting the two names are high-precision patterns then the name pair has a high confidence. The confidence is computed by the following formula.

$$Conf(N_i) = 1 - \prod_{j=1}^{k} (1 - \Pr ec(p_j)) \quad (1)$$

[3] Please refer to Grishman et al. (2005) and http://cs.nyu.edu/grishman/jet/license.html

Problem: While the intuition is correct, in practice this will over-rank NE pairs which are not only matched by patterns belonging to the target relation *R* but are also connected by patterns of many other relations. This is because of the initial settings used in many SSL systems: seeds are assigned high confidence. Thus all NE pairs matched by initial seed patterns will have very high confidence.

Suppose the target relation is *LocatedIn*, and "*visited*" is a seed pattern; then the <*Clinton*, *Arkansas*> example will be over-rated because we cannot take into account that it would also match patterns of other relations such as *PersonGovernorOfLocation* and *PersonBornInLocation* in a real corpus. This will cause a vicious circle, i.e., bogus NE pairs extract more bogus patterns which further extract more bogus NE pairs. We believe this flaw of the initial settings partially results in the semantic drift problem.

One can imagine that this is not a problem that can be solved by using a different formula to replace the one presented here. A possible solution is to study the structure of unlabeled data (NE pairs in our case) and integrate this structure information into the initial settings. Indeed, this is where pattern clusters come into play. We will demonstrate this in Section 3.2.

Step 2: Use NE pairs to search for new patterns and rank patterns.

Similar to the intuition in Step 1, for a pattern *p*, if many of the NE pairs it matches are very confident then *p* has many supporters and should have a high ranking. We can use formula (2) to estimate the confidence of patterns and rank them.

$$Conf(p) = \frac{Sup(p)}{|H|} \bullet \log Sup(p) \qquad (2)$$

Here $|H|$ is the number of unique NE pairs matched by *p* and *Sup(p)* is the sum of the support it can get from the $|H|$ pairs:

$$Sup(p) = \sum_{j=1}^{|H|} Conf(N_j) \qquad (3)$$

The precision of *p* is given by the average confidence of the NE pairs matched by *p*.

$$\Pr ec(p) = \frac{Sup(p)}{|H|} \qquad (4)$$

Formula (4) normalizes the precision to range from 0 to 1. As a result the confidence of each NE pair is also normalized to between 0 and 1.

Step 3: Accept patterns

Most systems accept the *K* top ranked patterns in Step 2 as new seeds, subject to some restrictions such as requiring the differences of confidence of the *K* patterns to be within a small range.

Step 4: Loop or stop

The procedure now decides whether to repeat from Step 1 or to terminate.

Most systems simply do not know when to stop. They either run a fixed number of iterations or use some held-out data to find one criterion that works the best for the held-out data.

3.2 Bootstrapping Guided by Clusters

Recall that our clustering algorithm in Section 2 provides us with *K* clusters, each of which contains *n* (*n* differs in different clusters) patterns. Every pattern in our corpus now has a cluster membership (the seed patterns have the same membership).

The most important benefit from our pattern clusters is that now we can measure how strongly a NE pair N_i is associated with our target cluster C_t (the one to which the seed patterns belong).

$$\Pr ob(N_i \in C_t) = \frac{\sum_{p \in C_t} freq(N_i, p)}{m} \qquad (5)$$

Here $freq(N_i, p)$ is the number of times *p* matches N_i and *m* is the total number of pattern instances matching N_i.

We integrate this prior cluster distribution of each NE pair into the initial settings of our new bootstrapping procedure.

Step1: Use seed patterns to match new NE pairs and evaluate NE pairs.

Assumption: A good NE pair must be strongly associated with the target cluster and can be matched by multiple high-precision patterns.

So we evaluate a NE pair by the harmonic mean of two confidence scores, namely the confidence as its association with the target cluster and the confidence given by the patterns matching it.

$$Conf(N_i) = 2 \bullet \frac{Semi\_Conf(N_i) \bullet Cluster\_Conf(N_i)}{Semi\_Conf(N_i) + Cluster\_Conf(N_i)} \quad (6)$$

$$Semi\_Conf(N_i) = 1 - \prod_{j=1}^{k} (1 - Prec(p_j)) \quad (7)$$

$$Cluster\_Conf(N_i) = \Pr ob(N_i \in C_t) \quad (8)$$

Under such settings, <*Clinton, Arkansas*> will be assigned a lower confidence score for the *LocatedIn* relation than it is in the standard bootstrapping. Even if we assign high precision to our seed patterns such as "*visited*" and consequently the *Semi_Conf* is very high, it can still be discounted by the *Cluster_Conf*[4].

Step 2: Use NE pairs to search for new patterns and rank patterns.

All the measurement functions are the same as those used in the standard bootstrapping. However, with better ranking of NE pairs in Step 1, the patterns are also ranked better than they are in the standard bootstrapping.

Step 3: Accept patterns

We also accept the *K* top ranked patterns.

Step 4: Loop or stop

Since each pattern in our corpus has a cluster membership, we can monitor the semantic drift easily and naturally stop: it drifts when the procedure tries to accept patterns which do not belong to the target cluster; we can stop when the procedure tends to accept more patterns outside of the target cluster.

If our clustering algorithm can give us perfect pattern clusters, we can stop bootstrapping immediately after it accepts the first pattern not belonging to the target cluster. Then the bootstrapping becomes redundant since all it does is to consume the patterns of the target cluster.

Facing the reality of the behavior of many clustering algorithms, we allow the procedure to occasionally accept patterns outside of the target cluster but we are not tolerant when it tries to accept more patterns outside of the target cluster than patterns in it. Note that when such patterns are accepted they will be moved to the target cluster and invoke the recomputation of *Cluster_Conf* of NE pairs connected by these patterns. The ranking functions in step 1 and 2 insure that the procedure will only accept patterns which can gain strong support from NE pairs that are strongly associated with the target cluster and are connected by many confident patterns.

4 Experiments

4.1 Corpus

Our corpora contain 37 years of news articles: TDT5, NYT(94-00), APW(98-00), XINHUA(96-00), WSJ(94-96), LATWP(94-97), REUFF(94-96), REUTE(94-96), and WSJSF(87-94). It contains roughly 65 million sentences and 1.3 billion tokens.

4.2 Seeds

Seeds of the 3 relations we are going to test are given in table 2. *LocatedIn* detects relation between PERSON and LOCATION/GPE; Social (*SOC*) detects social relations (either business or family) between PERSON and PERSON; Employment (*EMP*) detects employment relations between PERSON and ORGANIZATION.

Table 2. Seed Patterns

Relation	Seeds
Located-in	*nsubj'* visit *dobj* *nsubj'* travel *prep_to* *poss'* trip *prep_to*
SOC	*appos* friend/lawyer *poss* *appos* son/spokesman *prep_of/prep_for* *nsubj'* fire *dobj* *nsubjpass'* fire *agent*
EMP[5]	*appos* chairman/executive/founder *prep_of* *appos* editor *prep_of* *appos* director/head/officer/analyst *prep_at* *appos* manager *prep_with*

(*nsubj*, *dobj*, *prep*, *appos*, *poss*, *nsubjpass*, *agent* stand for subject, direct object, preposition, apposition, possessive, passive nominal subject and complement of passive verb. The quote marks in Table 2 and Table 3 denote inverse dependencies in the dependency path.)

We work on these three relations mainly because of the availability of benchmark evaluation data. These are the most frequent relations in our evaluation data.

[4] The *Cluster_Conf* of <*Clinton, Arkansas*> related to the *LocatedIn* relation is indeed very low (less than 0.1) in our experiments.

[5] We provide more seeds (executives and staff) for *EMP* because it has been pointed out in (Sun, 2009) that *EMP* contains a lot of job titles.

4.3 Unsupervised Experiments

We run the clustering algorithm described in Section 2 using all the 37 years' data. We require that a pattern match at least 7 distinct NE pairs and that an NE pair must be connected by at least 7 unique patterns. As a result, there are 635,128 patterns (22,225 unique ones) used in experiments. We use 0.005 as the cutoff threshold of complete linkage. The threshold is decided by trying a series of thresholds and searching for the maximal[6] one that is capable of placing the seed patterns for each relation into a single cluster. Table 3 shows the top 15 patterns (ranked by their corpus frequency) of the cluster into which our *LocatedIn* seeds fall.

Table 3. Top 15 patterns in the *LocatedIn* Cluster

Index	Pattern	Frequency
1	*nsubj'* said *prep_in*	2203
2	*nsubj'* visit *dobj*	1831
3	*poss'* visit *prep_to*	1522
4	*nsubj'* return *prep_to*	1394
5	*nsubj'* tell *prep_in*	1363
6	*nsubj'* be *prep_in*	1283
7	*nsubj'* arrive *prep_in*	1113
8	*nsubj'* leave *dobj*	1106
9	*nsubj'* go *prep_to*	926
10	*nsubj'* fly *prep_to*	700
11	*nsubj'* come *prep_to*	658
12	*appos* leader *poss*	454
13	*poss'* trip *prep_to*	442
14	*rcmod* be *prep_in*	419
15	*nsubj'* make *prep_in*	418

4.4 Semi-supervised Experiments

To provide strong statistical evidence, we divide our data into 10 folds (combinations of news articles from different years and different news resources). We then run both the standard and our new bootstrapping on the 10 folds. For both procedures, we accept n patterns in a single iteration (n is initialized to 2 and set to $n + 1$ after each iteration). We run 50 iterations in the standard bootstrapping and 1,325 patterns are accepted for each fold and each relation. Our new bootstrapping procedure stops when there are two consecutive iterations in which more than half of the newly accepted patterns do not belong to the target cluster. Thus the number of patterns accepted for each fold and each relation differs as the last iteration differs.

[6] We choose the maximal value because many clusters will be merged to a single one when the threshold is close to 0, making the clusters too general to be useful.

4.5 Evaluation

The output of our bootstrapping procedures is 60 sets of patterns (3 relations × 2 methods × 10 folds). We need a data set and evaluation method which can compare their effectiveness equally and consistently.

Evaluation data: ACE 2004 training data. ACE does not provide relation annotation between each pair of names. For example, in "*US President Clinton said that the United States …*" ACE annotates an *EMP* relation between the name "*US*" and nominal "*President*". There is no annotation between "*US*" and "*Clinton*". However, it provides entity co-reference information which connects "*President*" to "*Clinton*". So we take advantage of this entity co-reference information to automatically re-annotate the relations where possible to link a pair of names within a single sentence. The re-annotation yields an *EMP* relation between "*US*" and "*Clinton*". The re-annotation is reviewed by hand to avoid adding a relation linking "*Clinton*" and the more distant co-referent "*United States*", even though "*US*" and "*the United States*" refer to the same entity. This data set provides us with 412/3492 positive/negative relation instances between names. Among the 412 positive instances, there are 188/117/35 instances for *EMP/LocatedIn/SOC* relations.

Evaluation method: We adopt a direct evaluation method, i.e., use our sets of patterns to extract relations between names on ACE data. Applying patterns to a benchmark data set can provide us with better precision/recall analyses. We use a strict pattern match strategy. We can certainly take advantage of loose match or add patterns as additional features to feature-based relation extraction systems to boost our performance but we do not want these to complicate the comparison of the standard and our new bootstrapping procedures.

4.6 Results and Analyses

We average our results on the 10 folds. We plot precision against recall and semantic drift rate against iterations (Drift). We compute the semantic drift rate as the percentage of false

Figure 2. Performance for *EMP/LocatedIn/SOC*

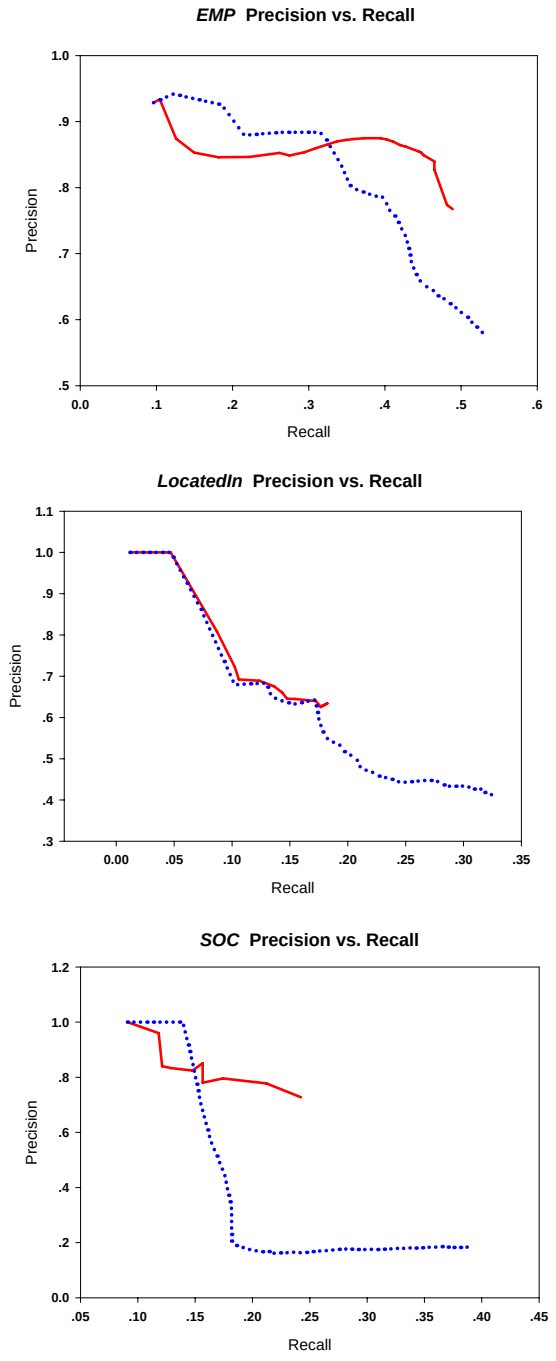
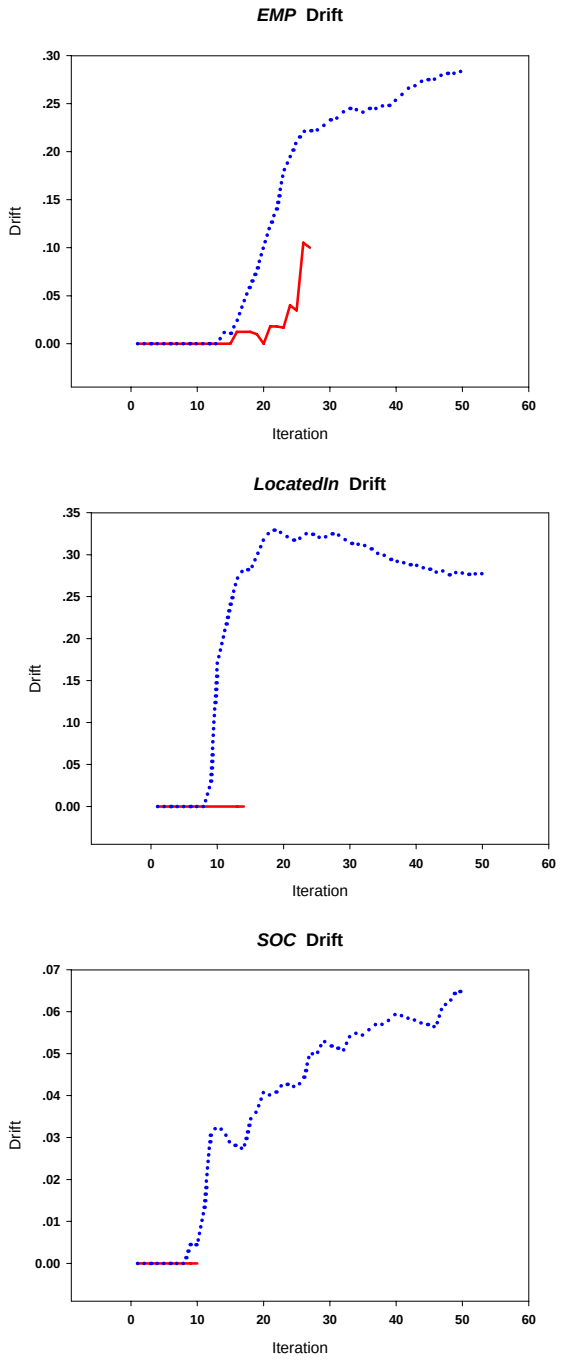


Figure 3. Drift for *EMP/LocatedIn/SOC*



positive instances belonging to ACE relations other than the target relation. Take *EMP* for example, we compute how many of the false positive instances belonging to other relations such as *LocatedIn*, *SOC* and other ACE relations. In all plots, red solid lines represent bootstrapping with guidance from clusters and blue dotted lines standard bootstrapping.

There are a number of conclusions that can be drawn from these results. We are particularly interested in the following two questions: To what extent did we prevent semantic drift by the guidance of pattern clusters? Did we stop at the right point, i.e., can we keep high precision while maintaining near maximal recall?

1) It is obvious from the drift curves that our bootstrapping effectively prevents semantic drift. Indeed, there is no drift at all when *LocatedIn*

and *SOC* learners terminate. Although drift indeed occurs in the *EMP* relation, its curve is much lower than that of the standard bootstrapping.

2) Our new procedure terminates when the precision is still high while maintaining a reasonable recall. Our bootstrapping for *EMP/SOC/LocatedIn* terminates at F-measures of 60/37/28 (in percentage). We conducted the Wilcoxon Matched-Pairs Signed-Ranks Test on the 10 folds, comparing the F-measures of the last iteration of our bootstrapping guided by clusters and the iteration which provides the best average F-measure over the 3 relations of the standard bootstrapping. The results show that the improvement of using clusters to guide bootstrapping is significant at a 97% confidence level.

We hypothesize that when working on dozens or hundreds of relations the gain of our procedure will be even bigger since we can effectively prevent inter-class errors.

5 Related Work

Recent research starts exploring unlabeled data for discriminative learning. Miller et al., (2004) augmented name tagging training data with hierarchical word clusters and encoded cluster membership in features for improving name tagging. Lin and Wu (2009) further explored a two-stage cluster-based approach: first clustering phrases and then relying on a supervised learner to identify useful clusters and assign proper weights to cluster features. Other similar work includes (Wong and Ng, 2007) for name tagging, and (Koo et. al., 2008) for dependency parsing.

While similar in spirit, our supervision is minimal, i.e., we only use a few seeds while the above approaches rely on a large amount of labeled data. To the best of our knowledge, the theme explored in this paper is the first study of using pattern clusters for preventing semantic drift in semi-supervised pattern discovery.

Recent research also explored the idea of driving SSL with explicit constraints constructed by hand such as identifying mutual exclusion of different categories (i.e., people and sport are mutually exclusive). This is termed constraint-driven learning in (Chang et al., 2007), coupled learning in (Carlson et al., 2010) and counter-training in (Yangarber, 2003). The learning quality largely depends on the completeness of explicit constraints. While we share the same goal, i.e., to prevent semantic drift, we rely on unsupervised clusters to discover implicit constraints for us instead of generating constraints by hand.

Our research is also close to semi-supervised IE pattern learners including (Riloff and Jones, 1999), (Agichtein and Gravano, 2000), (Yangarber et al., 2000), and many others. While they conduct bootstrapping on unlabeled data directly, we first cluster unlabeled data and then bootstrap with help from clusters.

There are also clear connections to work on unsupervised relation discovery (Hasegawa et al., 2004; Zhang et al., 2005; Rosenfeld and Feldman, 2007). They group pairs of names into relation clusters based on the contexts between names while we group the contexts/patterns into clusters based on features extracted from names.

6 Conclusions and Future Work

We presented a simple algorithm for clustering patterns and used pattern clusters to guide semi-supervised semantic pattern discovery. The novel bootstrapping procedure can achieve the best F-1 score while maintaining a good trade-off between precision and recall. We also demonstrated that it can effectively prevent semantic drift and naturally terminate.

We plan to extend this idea to improve relation extraction performance with a richer model as used in (Zhang et al., 2004; Zhou et al., 2008) than a simple pattern learner. The feature space will be much larger than the one adopted in this paper. We will investigate how to overcome the memory bottleneck when we apply rich models to millions of instances.

7 Acknowledgements

We would like to thank Prof. Satoshi Sekine for his useful suggestions.

References

Steven Abney. 2008. *Semisupervised Learning for Computational Linguistics*, Chapman and Hall.

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text

collections. In *Proc. of the Fifth ACM International Conference on Digital Libraries*.

Sergey Brin. Extracting patterns and relations from the World-Wide Web. 1998. In *Proc. of the 1998 Intl. Workshop on the Web and Databases*.

Razvan C. Bunescu and Raymond J. Mooney. 2005. A Shortest Path Dependency Kernel for Relation Extraction. In *Proc. of HLT/EMNLP*.

Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam Rafael Hruschka Junior and Tom M. Mitchell. 2010. Coupled Semi-Supervised Learning for Information Extraction. In *WSDM*.

Ming-Wei Chang, Lev Ratinov, and Dan Roth. 2007. Guiding semisupervision with constraint-driven learning. In *Proc. of ACL-2007*, Prague.

Zheng Chen and Heng Ji. 2009. Can One Language Bootstrap the Other: A Case Study on Event Extraction. In *NAACL HLT Workshop on Semi-supervised Learning for NLP*.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classication. In *Proc. of EMNLP-99*.

James R. Curran, Tara Murphy, and Bernhard Scholz. 2007. Minimising semantic drift with Mutual Exclusion Bootstrapping. In *Proc. of PACLING*.

Ralph Grishman, David Westbrook and Adam Meyers. 2005. NYU's English ACE 2005 System Description. ACE 2005 Evaluation Workshop.

Zellig S. Harris. 1954. Distributional Structure. *Word. Vol 10,1954, 146-162*.

Takaaki Hasegawa, Satoshi Sekine, Ralph Grishman. 2004. Discovering Relations among Named Entities from Large Corpora. In *Proc. of ACL-04*.

Marti Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proc. of the 14th Intl. Conf. on Computational Linguistics*.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple Semi-supervised Dependency Parsing. In *Proceedings of ACL-08: HLT*.

Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules for question-answering. *Natural Language Engineering, 7(4):343–360*.

Dekang Lin and Xiaoyun Wu. 2009. Phrase Clustering for Discriminative Learning. In *Proceedings of the ACL and IJCNLP 2009*.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The Stanford typed dependencies representation. In *COLING Workshop on Cross-framework and Cross-domain Parser Evaluation*.

Scott Miller, Jethran Guinness and Alex Zamanian. 2004. Name Tagging with Word Clusters and Discriminative Training. In *Proc. of HLT-NAACL*.

Patrick Pantel and Marco Pennacchiotti. 2006. Espresso: Leveraging Generic Patterns for Automatically Harvesting Semantic Relations. In *Proc. of COLING-06 and ACL-06*.

Deepak Ravichandran and Eduard Hovy. 2002. Learning Surface Text Patterns for a Question Answering System. In *Proc. of ACL-2002.*

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proc. of AAAI-99*.

Benjamin Rosenfeld, Ronen Feldman. 2007. Clustering for Unsupervised Relation Identification. In *Proc. of CIKM '07*.

Mark Stevenson and Mark A. Greenwood. 2006. Comparing Information Extraction Pattern Models. In *Proceedings of the Workshop on Information Extraction Beyond The Document*.

Mark Stevenson and Mark A. Greenwood. 2005. A Semantic Approach to IE Pattern Induction. In *Proc. of the 43rd Annual Meeting of the ACL*.

Ang Sun. 2009. A Two-stage Bootstrapping Algorithm for Relation Extraction. In *RANLP-09*.

Yingchuan Wong and Hwee Tou Ng. 2007. One Class per Named Entity: Exploiting Unlabeled Text for Named Entity Recognition. In *Proc. of IJCAI-07*.

Roman Yangarber. 2003. Counter-training in the discovery of semantic patterns. In *Proc. of ACL*.

Roman Yangarber, Ralph Grishman, Pasi Tapanainen and Silja Huttunen. 2000. Automatic acquisition of domain knowledge for information extraction. In *Proc. of COLING-2000*.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proc. of ACL-95*.

Min Zhang, Jian Su, Danmei Wang, Guodong Zhou, and Chew Lim Tan. 2005. Discovering Relations Between Named Entities from a Large Raw Corpus Using Tree Similarity-Based Clustering. In *IJCNLP 2005, LNAI 3651, pp. 378 – 389*.

Zhu Zhang. (2004). Weakly supervised relation classification for information extraction. In *Proc. of CIKM'2004*.

GuoDong Zhou, JunHui Li, LongHua Qian and QiaoMing Zhu. 2008. Semi-supervised learning for relation extraction. *IJCNLP'2008:32-39*.