Phrase Structure Parsing with Dependency Structure

Zhiguo Wang and **Chengqing Zong**
National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences
{zgwang, cqzong}@nlpr.ia.ac.cn

Abstract

In this paper we present a novel phrase structure parsing approach with the help of dependency structure. Different with existing phrase parsers, in our approach the inference procedure is guided by dependency structure, which makes the parsing procedure flexibly. The experimental results show our approach is much more accurate. With the help of golden dependency trees, F1 score of our parser achieves 96.08% on Penn English Treebank and 90.61% on Penn Chinese Treebank. With the help of N-best dependency trees generated by modified MSTParser, F1 score achieves 90.54% for English and 83.93% for Chinese.

1 Introduction

Over the past few years, several high-precision phrase parsers have been presented, and most of them are employing probabilistic context-free grammar (PCFG). As we all know, the basic PCFG has the problems that the independence assumption is too strong and lacks of lexical conditioning (Jurafsky and Martin, 2007). Although researchers have proposed various models and inference algorithms aiming to solve these problems, the performance of existing phrase parsers is still remained to further improve. Most of the existing approaches can be classified into two categories: unlexicalized PCFG based (Johnson, 1998; Klein and Manning, 2003; Levy and Manning, 2003; Matsuzaki et al., 2005; Petrov et al., 2006) and lexicalized PCFG based (Collins, 1999a; Charniak, 1997; Bikel, 2004; Charniak and Johnson, 2005).

Unlexicalized PCFG based approach attempts to weaken the independence assumption by annotating non-terminal symbols with labels of ancestor, siblings and even the latent annotations encoded by local information. In lexicalized PCFG based approach, researchers believe that the forms of a constituent and its sub-constituents are determined more by the constituent's head than any other of its lexical items (Charniak, 1997), so they annotate non-terminal symbols with the head words information.

Both of the two PCFG based approaches have improved the basic PCFG based parsers significantly. However, neither of them has been guided by enough linguistic priori knowledge. Their parsing procedures are too mechanical. Because of this, the efficiency is always worse, and much more artificial ambiguities, which are different from linguistic ambiguities (Krotov et al., 1998; Johnson, 1998), are generated. We believe parsing procedure guided by more linguistic priori knowledge will help to overcome the drawbacks in some extent. From our intuition, dependency structure, another type of syntactic structure with much linguistic knowledge, will be a good candidate to guide phrase parsing procedure.

In this paper we present a novel approach to using dependency structure to guide phrase parsing. This novel approach has its virtues from multiple angles. First, dependency structure offers a good compromise between the conflicting demands of analysis depth, which makes it much easier to get through hand annotating than phrase structure (Nivre, 2004). So, when we want to build a phrase structure corpus, we can build a dependency structure corpus first, and get the corresponding phrase structure automatically with the help of dependency structure. Second, many parsing algorithms with linear-time complexity used in dependency parsers can still achieve the state-of-the-art results (Johansson, 2007), but almost all phrase parsers with high-precision have no efficient algorithms superior to cubic-time complexity. So, in order to get an efficient parser, we can first get a dependency structure through linear-time algorithm, and then obtain the phrase structure with the help of dependency structure more efficiently. Third, the lexicalized PCFG based parsers which just bring the head words into account have got a highly improved performance. It gives us reasons to believe dependency structure which takes the relationship of all the words will bring phrase parser a great help.

Remainder of this paper is organized as follows: Section 2 introduces the related work. Section 3 gives a consistency between dependency structure and phrase structure, and presents an approach to parsing phrase structure with dependency structure. In Section 4, we discuss the experiments and analysis. Finally, we conclude this paper and point out some future work in Section 5.

2 Related work

Unlexicalized PCFG based parsers (Johnson, 1998; Klein and Manning, 2003; Levy and Manning, 2003; Matsuzaki et al., 2005; Petrov et al., 2006) are the most successful parsing tools. They regard parsing as a pure machine learning question. However, they haven't taken any extra linguistic priori knowledge directly into account. Lexicalized PCFG based parsers (Collins, 1999a; Charniak, 1997; Bikel, 2004; Charniak and Johnson, 2005) just bring a little linguistic priori knowledge (head word information) into learning phase. In inference phase, both of the unlexicalized PCFG based approach and lexicalized PCFG based approach are using the pure searching algorithms, which try to parse a sentence monotonously, either from left to right or from right to left. From these states, we can find that manners of current parsers are too mechanical. Because of this, the efficiency of phrase parsers is always worse, and much more artificial ambiguities are generated.

There have been some work (Collins et al., 1999b; Xia and Palmer, 2001) about converting dependency structures to phrase structures. Collins et al. (1999b) proposed an algorithm to convert the Czech dependency Treebank into a phrase structure Treebank and do dependency parsing through Collins (1999a)'s model. Results showed the accuracy of dependency parsing for Czech was improved largely. Xia and Palmer (2001) proposed a more generalized algorithm according to X-bar theory and Collins et al. (1999b), and they did some experiments on Penn Treebank. The results showed their algorithm produced phrase structures that were very close to the ones in Penn Treebank. However, we have to point out that they only computed the unlabeled performance but lost all the exact syntactic symbols. Different from tree-transformed PCFG based approach and lexicalized PCFG based approach, both of Collins et al. (1999b) and Xia and Palmer (2001) attempted to build some heuristic rules through linguistic theory, but didn't try to learn anything from Treebank.

Li and Zong (2005) presented a hierarchical parsing algorithm for long complex Chinese sentences with the help of punctuations. They first divided a long sentence into short ones according to punctuation marks, then parsed the short ones into sub-trees individually, and at last combined all the sub-trees into a whole tree. Experimental results showed the parsing time was reduced largely, and performance was improved too. Although the procedure of their parser is more close to human beings' manner, it appears a little shallow just using the punctuation marks.

In this paper our motivations are to bring more linguistic priori knowledge into phrase parsing procedure with the help of dependency structure, and make the parsing procedure flexibly.

Matsuzaki et al. (2005) defined a generative model called PCFG with latent annotations (PCFG-LA). Using EM-algorithm each non-terminal symbols was annotated with a latent variable, and a fine-grained model can be got. In order to get a more compact PCFG-LA model, Petrov et al. (2006) presented a split-and-merge method which can get PCFG-LA model hierarchically, and their final result outperformed state-of-the-art phrase parsers. To make the parsing process of hierarchical PCFG-LA model more efficient, Petrov and Klein (2007) presented a coarse-to-fine inference algorithm. In Section 4 of this paper, we try to combine the hierarchical PCFG-LA model in learning phase and coarse-to-fine method in inference phase into our parser in order to get an accurate and efficient parser.

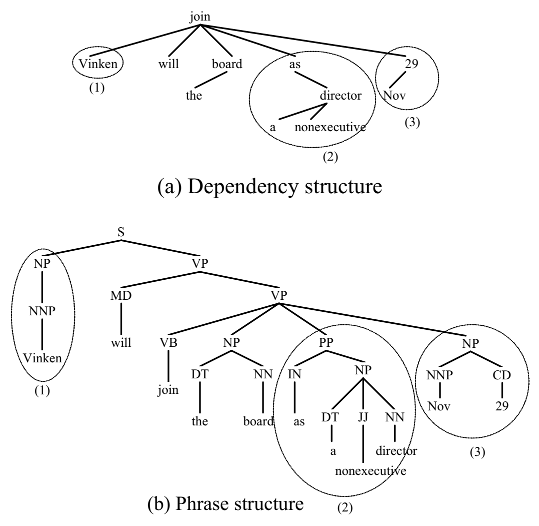


Figure 1. The consistency between phrase structure and dependency structure

3 Our framework

In this section, we first compare phrase structure with dependency structure of the same sentence, and get a consistent relationship among them. Then, based on this relationship, we present an inference framework to make the parsing procedure flexible and more efficient.

3.1 Analysis on consistency between phrase structure and dependency structure

Phrase structure and dependency structure are two different ways to represent syntactic structures of sentences. Phrase structure represents sentences by nesting of multi-word constituents, while dependency structure represents sentences as trees, whose nodes are words and edges represent the relations among words.

As we know, there are two kinds of dependency structures, projective structure and non-projective structure. For free-word order languages, non-projectivity is a common phenomenon, e.g. Czech. For languages like English and Chinese, the dependency structures are often projective trees. In this paper, we only consider English parsing based on Penn Treebank (PTB) and Chinese parsing based on Penn Chinese Treebank (PCTB), so we just research the consistency between phrase structure and projective dependency structure through PTB/PCTB.

Information carried by the two structures isn't equal. Phrase structure is more flexible, carries more information, and even contains all the information of dependency structure. So the task to convert a phrase structure to dependency structure is more straight, e.g. Nivre and Scholz (2004), Johansson and Nugues (2007). However, the reverse procedure is much more difficult, because dependency structure lacks the syntactic symbols, which are indispensable in phrase structure.

Although the two structures are completely different, they have consistency in some deep level. In this paper we analyze the consistency from a practical perspective in order to do phrase parsing with the help of dependency structure. Having investigated the two kinds of trees with dependency structure and phrase structure, we find a consistency[1] that each sub-tree in dependency structure must correspond to a sub-tree in phrase structure who dominates all the words appearing in dependency sub-tree. Figure 1 shows this relationship more intuitively. The dependency sub-tree surrounded by circle (1) in Figure 1(a) is a one-layer sub-tree, and has a corresponding phrase sub-tree surrounded by circle (1) in Figure 1(b). Both of the two sub-trees dominate the same word "Vinken". This consistency is also satisfied in other cases, e.g. two-layer sub-tree surrounded by circle (3) and three-layer sub-tree surrounded by circle (2) in Figure 1(a). These dependency sub-trees respectively have their corresponding phrase sub-trees dominating the same words in Figure 1(b).

This consistency brings us inspiration to make use of dependency structure for phrase parsing. In other words, in our method when a phrase sub-tree is generated from a dependency sub-tree, it must dominate all the same words with ones in the corresponding dependency sub-tree.

3.2 Inference framework

[1] Be aware that the consistency is irreversible and not every phrase sub-tree has its corresponding dependency sub-tree.

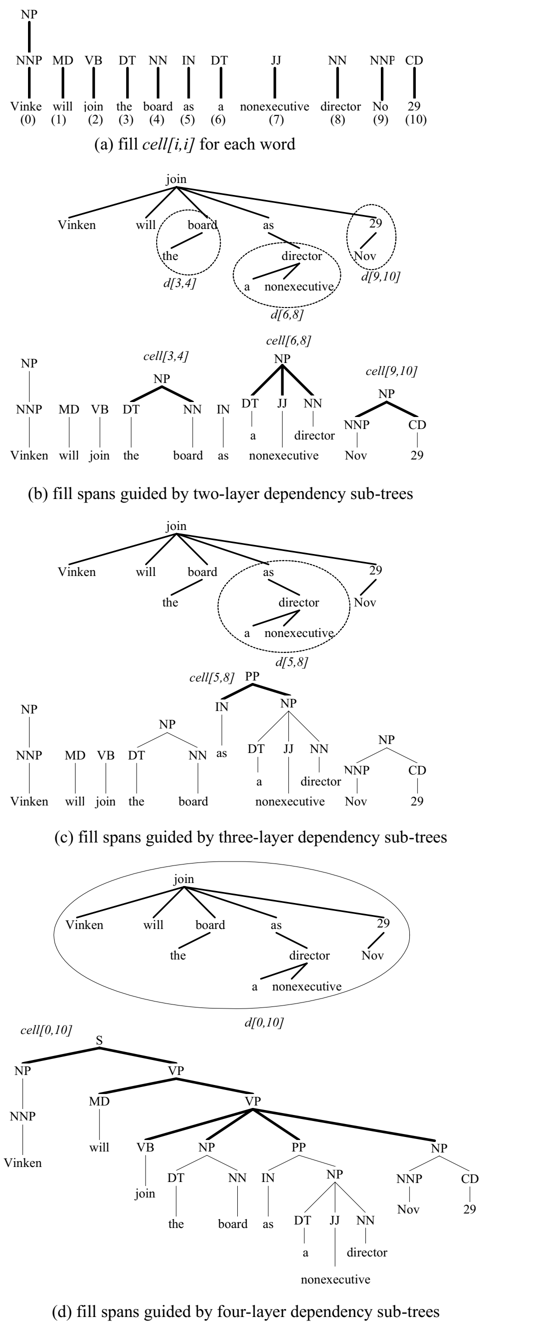

(a) fill cell[i,i] for each word
(b) fill spans guided by two-layer dependency sub-trees
(c) fill spans guided by three-layer dependency sub-trees
(d) fill spans guided by four-layer dependency sub-trees
d[3,4]
d[6,8]
d[9,10]
d[5,8]
d[0,10]
cell[3,4]
cell[6,8]
cell[9,10]
cell[5,8]
cell[0,10]

Figure 2. Parsing procedure of our inference framework guided by dependency structure

As we mentioned in Section 2, most of current inference algorithms are monotonous, which generate much more artificial ambiguities. For example, in Figure 1, if a sub-tree only dominating "board" and "as" is built (actually it is not occurred in golden tree) an artificial ambiguity is generated, and it thus will further bring a worse effect to other parts. The final precision will certainly descend. However, if we are given a corresponding dependency structure, those errors will be avoided. The consistency analyzed above tells us that there isn't a sub-tree dominating only "board" and "as" in dependency tree, so the two words can't build a sub-tree independently in phrase parsing. According to this strategy, we design an inference framework for phrase parsing.

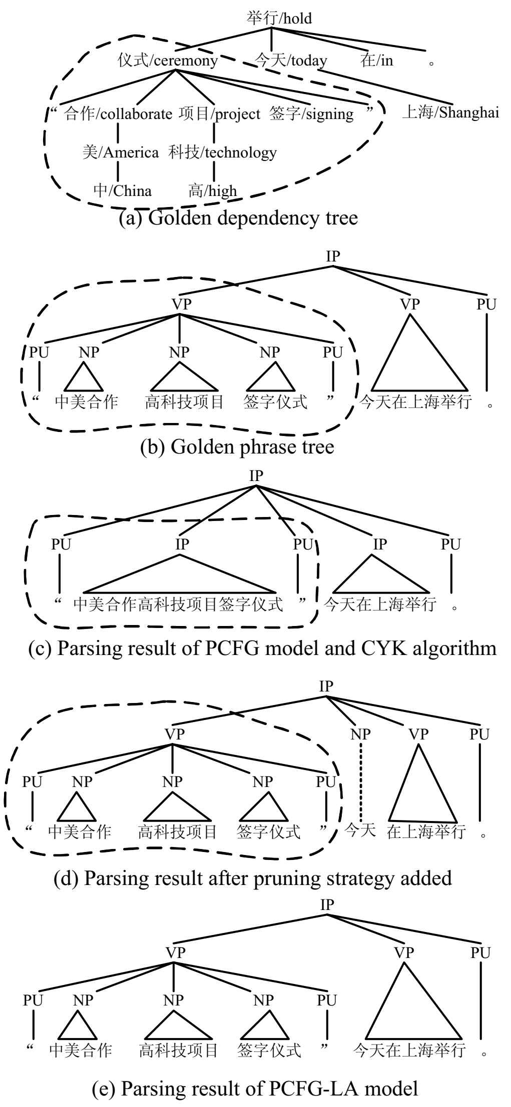
Our inference framework parses a sentence flexibly with a traditional inference algorithm. The following terms will help to explain our work. A key data structure is *cell[i,j]*, which is used to store phrase sub-trees spanning words from positions i to j of the input sentence. *d[i,j]* is a dependency sub-tree spanning words from positions i to j. *cells[i,j]* is an array to store all the *cell*s which can be combined to build *cell[i,j]*. The pseudo-code of our inference framework is shown in Algorithm1. The line indicated by (1) and (2) gives us freedom to select any kinds of inference algorithms and matching parsing models.

```
Algorithm 1
InferenceFramework(sentence S, dependency tree D)
----------------------------------------------------
▷ initialize a List for the input sentence
for each word w_i in S do
      fill cell[i, i] and add it to a list L

▷ parse the cells[] hierarchically
for each d[s, t] of D in topological order do
      fill cells[s, t] with spans in L
      fill cell[s, t] with cells[s, t] through
               traditional inference algorithm   (1)
      add cell[s, t] to L

▷ extract the best tree
estimate all trees in cell[0, n]
                through parsing model            (2)
return the best phrase tree
```

Now, let's illustrate the flexible parsing procedure step by step through an example. Please see Figure 2. For simplicity, we just draw sub-trees of the final best tree, and ignore all the others. Figure 2(a) shows the procedure of filling *cell[i,i]* for each word. In Figure 2(b), there are three two-layer dependency sub-trees *d[3,4]*, *d[6,8]* and *d[9,10]*. So we try to generate phrase sub-trees for *cell[3,4]*, *cell[6,8]* and *cell[9,10]*, which have been annotated with bold edges. For example, we use sub-trees contained in *cell[6,6]*, *cell[7,7]* and *cell[8,8]* to



举行/hold
仪式/ceremony
今天/today
在/in
。
“ 合作/collaborate 项目/project 签字/signing ”
上海/Shanghai
美/America 科技/technology
中/China 高/high
(a) Golden dependency tree
IP VP VP PU
PU NP NP NP PU
“ 中美合作 高科技项目 签字仪式 ” 今天在上海举行 。
(b) Golden phrase tree
IP PU IP PU IP PU
“ 中美合作高科技项目签字仪式 ” 今天在上海举行 。
(c) Parsing result of PCFG model and CYK algorithm
IP VP NP VP PU
PU NP NP NP PU
“ 中美合作 高科技项目 签字仪式 ” 今天 在上海举行 。
(d) Parsing result after pruning strategy added
IP VP VP PU
PU NP NP NP PU
“ 中美合作 高科技项目 签字仪式 ” 今天在上海举行 。
(e) Parsing result of PCFG-LA model

Figure 4. An example showing experimental results

build new sub-trees for *cell[6,8]*. Figure 2(c) and Figure 2(d) show the same procedure for parsing with the help of three-layer dependency sub-trees and four-layer dependency sub-trees individually. The generated phrase sub-trees are all annotated with bold edges in the figure. Obviously, the biggest dependency sub-tree is the whole dependency tree of sentence. In this example, when the four-layer dependency sub-tree is processed, the whole phrase trees are built. Usually, more than one phrase trees with the similar skeletons are generated. So we use a model to evaluate candidate results, and get out the one with the highest score as the final result.

Benefiting from the dependency structure, we can parse a sentence flexibly. Comparing with previous work on converting dependency structure to phrase structure (Collins et al., 1999b; Xia and Palmer, 2001), we make use of Treebank knowledge more sufficient with the help of traditional parsing technology. The search space has been pruned tremendously. As we know, the traditional parsing approach often tries to search all the n*(n+1)/2 cells for input sentence which has n words, but our parsing framework search cells intelligently with the help of corresponding dependency structure. Let's get a view of this through the sentence shown in Figure 2. From the whole parsing procedure shown in Figure 2, our framework just tries to fill 16 cells, which are *cell[i,i]* for each word, *cell[3,4]*, *cell[6,8]*, *cell[9,10]*, *cell[5,8]* and *cell[0,10]* hierarchically, but traditional parsing approach would try to fill all 66 cells. So 75.76% searching space has been pruned.

	English	Chinese
Train Set	Sections 2-21	Art. 1-270, 400-1151
Dev Set	Section 22	Articles 301-325
Test Set	Section 23	Articles 271-300

Table 1. Experimental settings

4 Experiments and results

In order to evaluate the effectiveness of our approach, we have done some experiments both for English parsing and Chinese parsing.

4.1 Preparation

To make comparison with previous work fairly, our experiments are based on general Treebank according to standard settings. We choose Penn English Treebank for English parsing experiments and Penn Chinese Treebank for Chinese. Table 1 shows the standard settings we take.

Because the two Treebanks are in type of phrase structure, we should get dependency structures corresponding with them. There are two ways to accomplish this work. First, use converting tools to get dependency trees directly through converting the original Treebanks, and the generated trees are always considered as golden trees during dependency parsing. Second, use a dependency parser with state-of-the-art performance to parse all the sentences automatically. In this paper, we design two groups of experiments, as following:

(1) Phrase parsing with the help of golden dependency trees.
(2) Phrase parsing with the help of N-best dependency trees generated automatically.

4.2 Phrase parsing with golden dependency trees

In order to verify how much dependency structure can help phrase parsing and get an upper bound of our approach, we do phrase parsing with the help of golden dependency trees in this subsection.

Based on the parsing framework shown in Figure 3, we only use the basic PCFG in learning phase and our inference framework with basic CYK algorithm in inference phase. The parsing results are shown with the mark (1) in Table 2 for English and Table 3 for Chinese respectively.

Having investigated the generated trees with golden trees, we find the consistency of dependency structure and phrase structure is broken by some trees. Let's get a view of this through an example from Penn Chinese Treebank. In Figure 4(a), the dependency sub-tree surrounded by circle tells us that there must be a phrase sub-tree which dominate the word sequence of "中 美 合作 高 科技 项目 签字 仪式" (the signing ceremony of collaborating in high technology between America and China), and the golden phrase tree shown in Figure 4(b) has a corresponding sub-tree surrounded by circle indeed. However, the parsing tree generated by our approach shown in Figure 4(c) doesn't conform. There are three sub-trees dominating the word sequence mutually, but they don't construct a whole one. In our opinion, the contradiction derived from binarizing process of CYK[2]. The binary trees generated by our algorithm have consisted with the consistency originally, but after debinarizing process the consistency is broken.

Trying to check our opinion, we add a pruning strategy to the original inference algorithm to prune all the medial nodes which may break the consistency during parsing procedure. With the help of pruning strategy, the performances of English and Chinese are all improved further. Corresponding figures are shown in Table 2 and Table 3 with the mark (2). The parsing result of above example is shown in Figure 4(d) and the error appearing in Figure 4(c) is corrected naturally after the pruning strategy added.

Comparing with previous work which have done much work in learning phase, our algorithm achieves such amazing results only using basic PCFG model. From this aspect, our inference framework is much more effective. However, there are still some errors our approach can't deal with. For example, in Figure 4(d) the sub-tree rooted at NP and dominating word sequence of "今天 在 上海 举行" (hold in Shanghai today) is separated by two sub-trees. The reason is that the model (basic PCFG) we use in learning phase is too coarse to disambiguate sufficiently. So we don't pin all hopes in inference phase. We also modify the model in learning phase. PCFG-LA is one of the most successful models in phrase parsing, so we choose PCFG-LA as the model in learning phase. After this modification, performance of our approach has been improved delightedly. F1 score is 96.08% for English and 90.06% for Chinese. The line marked with (3) in Table 2 and Table 3 shows more details.

4.3 Phrase parsing with N-best dependency trees generated automatically

The experimental results shown in subsection 4.2 bring us confidence that do phrase parsing with the help of dependency structure is a highly effective approach. However, we don't usually have golden dependency structures, and a more acceptable way is using a dependency parser to generate dependency trees automatically. In this subsection we explore feasibility and effectiveness of phrase parsing with the help of dependency trees generated automatically. As we all know, even state-of-the-art dependency parser cannot generate totally correct result. So in order to make our system more robust we use N-best dependency structures to guide phrase parsing procedure.

[2] The premise of using CYK is that all the rules must have CNF form. So we usually bring some medial nodes to binarize rules gathered from Treebank.

	length<=40			all sentences		
	Precision	Recall	F1	Precision	Recall	F1
(1) Using PCFG and CYK	90.28	88.41	89.34	90.11	88.32	89.21
(2) Using pruning strategy	90.69	89.53	90.11	90.51	89.45	89.97
(3) Using PCFG-LA	96.28	95.97	96.13	96.25	95.91	96.08

Table 2. Parsing performance (%) for English with the help of golden dependency tree.

	length<=40			all sentences		
	Precision	Recall	F1	Precision	Recall	F1
(1) Using PCFG and CYK	86.89	78.25	82.34	85.56	77.43	81.29
(2) Using pruning strategy	87.65	82.33	84.91	86.39	81.45	83.85
(3) Using PCFG-LA	91.51	91.26	91.38	90.43	90.79	90.61

Table 3. Parsing performance (%) for Chinese with the help of golden dependency tree.

We choose MSTParser[3] which is the most famous dependency parser and modify it to generate N-best dependency trees. The oracle unlabeled accuracy of N-best dependency trees generated from 1-order model is shown in Table 4. To show the effectiveness of our approach, we choose Berkeleyparser[4] as the baseline parser, take the same configuration and combine it into our general parsing framework shown in Figure 3.

Considering the number of dependency structures (N-best) will affect the final result, we make use of the development set shown in Table1 to turning parameters. We parse the development set many times with different number of dependency structures. The F1 scores are shown in Figure 5 for English and Figure 6[5] for Chinese. From Figure 5 and Figure 6, we can find when we use 10-best dependency structures the performance is better. So we choose 10-best dependency trees for the test set.

The final performances of test set comparing with previous work are shown in Table 5 and Table 6. We can easily find that our approach has outperformed all the parsers which aren't improved through reranking stage or semi-supervised approach. Although there is still a margin between our parser and reranked parser or semi-supervised parser, we believe that the parsing performance can be improved further if we bring the reranking or semi-supervised approaches into our parsing framework.

4.4 Discussion

The experiment of parsing with golden dependency structure gets an amazing performance. It brings us a new way to build PTB/PCTB style phrase structure corpus. Because dependency structure is much easier to get through hand annotating than phrase structure, we can build a dependency structure corpus first, and then get phrase structure corpus through our approach guided by the dependency structure corpus.

The experiment of parsing with N-best dependency structures generated automatically uplifts the parsing performance to a new height. It brings us a more applied parsing tool for other NLP applications.

From the experiments in Section 4.2, we can find that even using the golden dependency structure we can't get totally correct phrase structure. The reason is that although every dependency sub-tree has its corresponding phrase sub-tree, not every phrase sub-tree has its corresponding dependency sub-tree. So the remainder errors can't be solved only by dependency structure and a better way is to modify the parsing model.

5 Conclusion and Future work

In this paper, we present a novel phrase parsing approach with the help of dependency structure. Based on the consistency between phrase structure and dependency structure, we propose a novel inference framework. Guided by the inference framework, inference algorithms parse sentences hierarchically with the help of dependency structures. Experimental results show that our approach can efficiently get better performance with both golden dependency structure and N-best dependency

[3] http://www.ryanmcd.com/MSTParser/MSTParser.html
[4] http://code.google.com/p/berkeleyparser/
[5] F1 score at n=0 is the result of Berkeley parser running on my machine

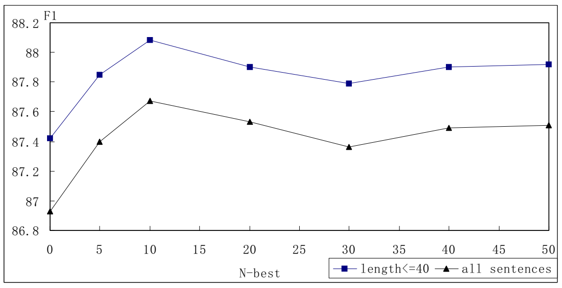


Figure 5. F1 scores (%) of Dev Set for English with the help of N-best dependency trees

	English		Chinese	
	len<=40	all	len<=40	all
5-best	90.62	90.49	87.92	84.93
10-best	91.6	91.48	89.05	85.9
20-best	92.36	92.21	89.86	86.79
30-best	92.74	92.6	90.3	87.28
40-best	92.96	92.83	90.62	87.63
50-best	93.08	92.95	90.79	87.87

Table 4. Oracle unlabeled accuracy (%) of N-best dependency structures generated from MSTParser

	<=40	All
Collins(1999)	88.6	88.2
Charniak and Johnson(2005)	90.1	89.55
Petrov and Klein(2007)	90.6	90.05
This Paper	**91.13**	**90.54**
Reranked		
Charniak and Johnson(2005)	92.0	91.4
Huang(2008)	——	**91.7**
Semi-supervised		
McClosky et al. (2006)	——	**92.1**

Table 5. F1 (%) of Test Set for English of previous work and our approach

	<=40	All
Chiang et al.(2002)	79.93	76.57
Bikel Thesis(2004)	81.2	79.0
Petrov and Klein(2007)	86.3	83.32
This Paper	**86.76**	**83.93**
Semi-supervised		
Huang and Harper(2009)	——	**85.18**

Table 6. F1 (%) of Test Set for Chinese of previous work and our approach

structures generated automatically.

However, there are still some problems remaining to further study. First, in our approach we just use the unlabeled dependency trees. The relationship labels carry some useful information too, and we can make use of them to further improve phrase parsing. Second, phrase structure can also help the process of dependency parsing (McDonald et al., 2006), so

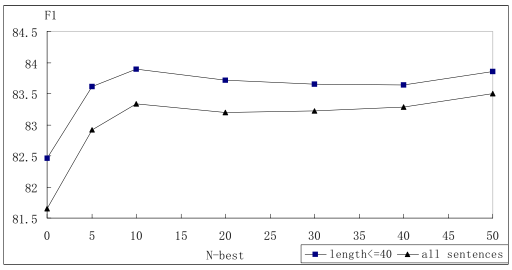


Figure 6. F1 scores (%) of Dev Set for Chinese with the help of N-best dependency trees

we can combine phrase parsing process and dependency parsing process together and make them help each other.

Acknowledgments

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

Alexander Krotov, Mark Hepple, Robert Gaizauskas, and Yorick Wilks. 1998. *Compacting the Penn Treebank grammar*. In ACL-COLING '98, pages 699-703.

Dan Klein and Chris Manning. 2003. *Accurate Unlexicalized Parsing*, In ACL '03, pages 423-430.

Daniel Jurafsky and James H. Martin. 2007. *SPEECH and LANGUAGE PROCESSING*--a draft, at Chapter 14.4.

Daniel M. Bikel. 2004. *On the Parameter Space of Generative Lexicalized Statistical Parsing Models*. Ph.D. thesis, U. of Pennsylvania.

Daniel M. Bikel. 2004. *Intricacies of Collins' Parsing Model*. In Computational Linguistics, 30(4), pages 479-511.

Daniel M. Bikel and David Chiang. 2000. *Two Statistical Parsing Models Applied to the Chinese Treebank*. In the Proceedings of the Second Chinese Language Processing Workshop.

David Chiang and Daniel M. Bikel. 2002. *Recovering Latent Information in Treebanks*. In COLING '02.

David McClosky, Eugene Charniak and Mark Johnson. 2006. *Effective self-training for parsing*. In ACL-06.

Deyi Xiong, Qun Liu and Shouxun Lin. 2005. *Lexicalized Beam Thresholding Parsing with Prior and Boundary Estimates*. the 6th Conference on Intelligent Text Processing and Computational Linguistics (CICLing), Pages 132 – 141.

D.H. Younger. 1967. *Recognition and parsing of context-free-languages in time n^3*. Information and Control, 10(2):189-208.

Eugene Charniak. 1997. *Statistical parsing with a context-free grammar and word statistics*. Proceedings of the Fourteenth National Conference on Artificial Intelligence AAAI Press/MIT Press, Menlo Park.

Eugene Charniak. 2000. *A maximum-entropyinspired parser*. In NAACL '00, pages 132–139.

Eugene Charniak and Mark Johnson. 2005. *Coarse-to-Fine n-Best Parsing and MaxEnt Discriminative Reranking*. In ACL '05.

Fei Xia and Martha Palmer. 2001. *Converting Dependency Structures to Phrase Structures*. The 1st Human Language Technology Conference (HLT-2001).

H. Gaifman. 1965. *Dependency Systems and phrase-Structure Systems*. Information and Control, pages 304-337.

H. Yamada and Y. Matsumoto. 2003. *Statistical dependency analysis with support vector machines*. In Proceedings of IWPT

J. Nivre, M. Scholz. 2004. *Deterministic dependency parsing of English text*. In COLING '04.

Liang Huang. 2008. *Forest reranking: Discriminative parsing with non-local features*. In ACL '08.

Mark Johnson. 1998. *PCFG models of linguistic tree representations*. Computational Linguistics, 24(4):613–632.

Michael Collins. 1999a. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, U. of Pennsylvania.

Michael Collins, Jan Hajic, Lance Ramshaw and Christoph Tillmann. 1999b. *A Statistical Parser for Czech*. In ACL '99.

Richard Johansson and Pierre Nugues. 2007. *Extended Constituent-to-dependency Conversion for English*. In Proceedings of NODALIDA.

Roger Levy, Christopher Manning. 2003. *Is it harder to parse Chinese, or the Chinese Treebank?* In ACL '03.

Ryan McDonald, Koby Grammer and Fernando Pereira. 2006. *Online learning of approximate dependency parsing algorithms*. In EACL '06.

Slav Petrov and Dan Klein. 2007. *Improved Inference for Unlexicalized Parsing*. In HLT-NAACL '07.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. *Learning Accurate, Compact, and Interpretable Tree Annotation*. In COLING-ACL '06.

T. Matsuzaki, Y. Miyao, and J. Tsujii. 2005. *Probabilistic CFG with latent annotations*. In ACL '05, pages 75–82.

T. Kasami. 1965. *An efficient recognition and syntax analysis algorithm for context-free languages*. Technical Report, AFCRL-65-758, Air Force Cambridge Reserch Lab., Bedford, MA

Xavier Carreras, Michael Collins, and Terry Koo. *TAG, Dynamic Programming and the Perceptron for Efficient, Feature-rich Parsing*. In CONLL '08.

Xing Li, Chengqing Zong. 2005. *A Hierarchical Parsing Approach with Punctuation Processing for Long Complex Chinese Sentences*. In IJCNLP '05

Yusuke Miyao, Rune Sætre, Kenji Sagae, Takuya Matsuzaki and Jun'ichi Tsujii. 2008. *Task-oriented Evaluation of Syntactic Parsers and Their Representations*. In ACL '08, pages 46-54.

Zhongqiang Huang, Mary Harper. 2009. *Self-Training PCFG Grammars with Latent Annotations Across Languages*. In EMNLP '09.