Improving Dependency Parsing with Subtrees from Auto-Parsed Data

Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa
Language Infrastructure Group, MASTAR Project
National Institute of Information and Communications Technology
3-5 Hikari-dai, Seika-cho, Soraku-gun, Kyoto, Japan, 619-0289
{chenwl, kazama, uchimoto, torisawa}@nict.go.jp

Abstract

This paper presents a simple and effective approach to improve dependency parsing by using subtrees from auto-parsed data. First, we use a baseline parser to parse large-scale unannotated data. Then we extract subtrees from dependency parse trees in the auto-parsed data. Finally, we construct new subtree-based features for parsing algorithms. To demonstrate the effectiveness of our proposed approach, we present the experimental results on the English Penn Treebank and the Chinese Penn Treebank. These results show that our approach significantly outperforms baseline systems. And, it achieves the best accuracy for the Chinese data and an accuracy which is competitive with the best known systems for the English data.

1 Introduction

Dependency parsing, which attempts to build dependency links between words in a sentence, has experienced a surge of interest in recent times, owing to its usefulness in such applications as machine translation (Nakazawa et al., 2006) and question answering (Cui et al., 2005). To obtain dependency parsers with high accuracy, supervised techniques require a large amount of hand-annotated data. While hand-annotated data are very expensive, large-scale unannotated data can be obtained easily. Therefore, the use of large-scale unannotated data in training is an attractive idea to improve dependency parsing performance.

In this paper, we present an approach that extracts subtrees from dependency trees in auto-parsed data to improve dependency parsing. The auto-parsed data are generated from large-scale unannotated data by using a baseline parser. Then, from dependency trees in the data, we extract different types of subtrees. Finally, we represent subtree-based features on training data to train dependency parsers.

The use of auto-parsed data is not new. However, unlike most of the previous studies (Sagae and Tsujii, 2007; Steedman et al., 2003) that improved the performance by using entire trees from auto-parsed data, we exploit partial information (i.e., subtrees) in auto-parsed data. In their approaches, they used entire auto-parsed trees as newly labeled data to train the parsing models, while we use subtree-based features and employ the original gold-standard data to train the models. The use of subtrees instead of complete trees can be justified by the fact that the accuracy of partial dependencies is much higher than that of entire dependency trees. Previous studies (McDonald and Pereira, 2006; Yamada and Matsumoto, 2003; Zhang and Clark, 2008) show that the accuracies of complete trees are about 40% for English and about 35% for Chinese, while the accuracies of relations between two words are much higher: about 90% for English and about 85% for Chinese. From these observations, we may conjecture that it is possible to conduct a more effective selection by using subtrees as the unit of information.

The use of word pairs in auto-parsed data was tried in van Noord (2007) and Chen et al. (2008). However, the information on word pairs is limited. To provide richer information, we consider more words besides word pairs. Specifically, we use subtrees containing two or three words extracted from dependency trees in the auto-parsed data. To demonstrate the effectiveness of our proposed approach, we present experimental results on English and Chinese data. We show that this simple approach greatly improves the accuracy and that the use of richer structures (i.e, word triples) indeed gives additional improvement. We also demonstrate that our approach and other improvement techniques (Koo et al., 2008; Nivre and McDonald, 2008) are complementary and that we can achieve very high accuracies when we combine our method with other improvement techniques. Specifically, we achieve the best accuracy for the Chinese data.

The rest of this paper is as follows: Section 2 introduces the background of dependency parsing. Section 3 proposes an approach for extracting subtrees and represents the subtree-based features for dependency parsers. Section 4 explains the experimental results and Section 5 discusses related work. Finally, in section 6 we draw conclusions.

2 Dependency parsing

Dependency parsing assigns head-dependent relations between the words in a sentence. A simple example is shown in Figure 1, where an arc between two words indicates a dependency relation between them. For example, the arc between "ate" and "fish" indicates that "ate" is the head of "fish" and "fish" is the dependent. The arc between "ROOT" and "ate" indicates that "ate" is the ROOT of the sentence.

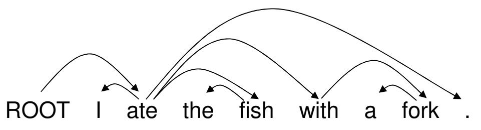


Figure 1: Example for dependency structure

2.1 Parsing approach

For dependency parsing, there are two main types of parsing models (Nivre and McDonald, 2008): graph-based model and transition-based model, which achieved state-of-the-art accuracy for a wide range of languages as shown in recent CoNLL shared tasks (Buchholz et al., 2006; Nivre et al., 2007). Our subtree-based features can be applied in both of the two parsing models.

In this paper, as the base parsing system, we employ the graph-based MST parsing model proposed by McDonald et al. (2005) and McDonald and Pereira (2006), which uses the idea of Maximum Spanning Trees of a graph and large margin structured learning algorithms. The details of parsing model were presented in McDonald et al. (2005) and McDonald and Pereira (2006).

2.2 Baseline Parser

In the MST parsing model, there are two well-used modes: the first-order and the second-order. The first-order model uses first-order features that are defined over single graph edges and the second-order model adds second-order features that are defined on adjacent edges.

For the parsing of unannotated data, we use the first-order MST parsing model, because we need to parse a large number of sentences and the parser must be fast. We call this parser the Baseline Parser.

3 Our approach

In this section, we describe our approach of extracting subtrees from unannotated data. First, we preprocess unannotated data using the Baseline Parser and obtain auto-parsed data. Subsequently, we extract the subtrees from dependency trees in the auto-parsed data. Finally, we generate subtree-based features for the parsing models.

3.1 Subtrees extraction

To ease explanation, we transform the dependency structure into a more tree-like structure as shown in Figure 2, the sentence is the same as the one in Figure 1.

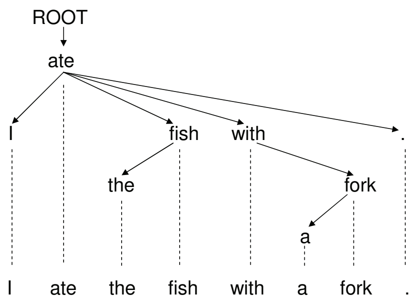


Figure 2: Example for dependency structure in tree-format

Our task is to extract subtrees from dependency trees. If a subtree contains two nodes, we call it a bigram-subtree. If a subtree contains three nodes, we call it a trigram-subtree.

3.2 List of subtrees

We extract subtrees from dependency trees and store them in list L_{st}. First, we extract bigram-subtrees that contain two words. If two words have a dependency relation in a tree, we add these two words as a subtree into list L_{st}. Similarly, we can extract trigram-subtrees. Note that the dependency direction and the order of the words in the original sentence are important in the extraction. To enable this, the subtrees are encoded in the string format that is expressed as $st = w : wid : hid(-w : wid : hid)+$[1], where w refers to a word in the subtree, wid refers to the ID (starting from 1) of a word in the subtree (words are ordered according to the positions of the original sentence)[2] , and hid refers to an ID of the head of the word (hid=0 means that this word is the root of a subtree). For example, "ate" and "fish" have a right dependency arc in the sentence shown in Figure 2. So the subtree is encoded as "ate:1:0-fish:2:1". Figure 3 shows all the subtrees extracted from the sentence in Figure 2, where the subtrees in (a) are bigram-subtrees and the ones in (b) are trigram-subtrees.

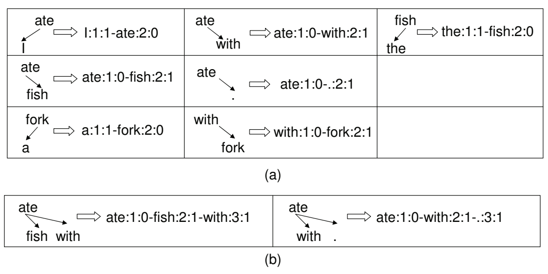


Figure 3: Examples of subtrees

Note that we only used the trigram-subtrees containing a head, its dependent $d1$, and $d1$'s leftmost right sibling[3]. We could not consider the case where two children are on different sides[4] of the head (for instance, "I" and "fish" for "ate" in Figure 2). We also do not use the child-parent-grandparent type (grandparent-type in short) trigram-subtrees. These are due to the limitations of the parsing algorithm of (McDonald and Pereira, 2006), which does not allow the features defined on those types of trigram-subtrees.

We extract the subtrees from the auto-parsed data, then merge the same subtrees into one entry, and count their frequency. We eliminate all subtrees that occur only once in the data.

[1] $+$ refers to matching the preceding element one or more times and is the same as a regular expression in Perl.

[2] So, wid is in fact redundant but we include it for ease of understanding.

[3] Note that the order of the siblings is based on the order of the words in the original sentence.

[4] Here, "side" means the position of a word relative to the head in the original sentence.

3.3 Subtree-based features

We represent new features based on the extracted subtrees and call them subtree-based features. The features based on bigram-subtrees correspond to the first-order features in the MST parsing model and those based on trigram-subtrees features correspond to the second-order features.

We first group the extracted subtrees into different sets based on their frequencies. After experiments with many different threshold settings on development data sets, we chose the following way. We group the subtrees into three sets corresponding to three levels of frequency: "high-frequency (HF)", "middle-frequency (MF)", and "low-frequency (LF)". HF, MF, and LF are used as set IDs for the three sets. The following are the settings: if a subtree is one of the TOP-10% most frequent subtrees, it is in set HF; else if a subtree is one of the TOP-20% subtrees, it is in set MF; else it is in set LF. Note that we compute these levels within a set of subtrees with the same number of nodes. We store the set ID for every subtree in L_{st}. For example, if subtree "ate:1:0-with:2:1" is among the TOP-10%, its set ID is HF.

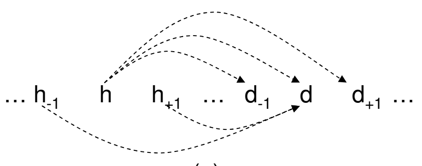
3.3.1 First-order subtree-based features

The first-order features are based on bigram-subtrees that are related to word pairs. We generate new features for a head h and a dependent d in the parsing process. Figure 4-(a)[5] shows the words and their surrounding words, where h_{-1} refers to the word to the left of the head in the sentence, h_{+1} refers to the word to the right of the head, d_{-1} refers to the word to the left of the dependent, and d_{+1} refers to the word to the right of the dependent. Temporary bigram-subtrees are formed by word pairs that are linked by dashed-lines in the figure. Then we retrieve these subtrees in L_{st} to get their set IDs (if a subtree is not included in L_{st}, its set ID is ZERO. That is, we have four sets: HF, MF, LF, and ZERO.).

Then we generate first-order subtree-based features, consisting of indicator functions for set IDs of the retrieved bigram-subtrees. When generating subtree-based features, each dashed line in Figure 4-(a) triggers a different feature.

To demonstrate how to generate first-order subtree-based features, we use an example that is as follows. Suppose that we are going to parse the sentence "He ate the cake with a fork." as shown

[5] Please note that d could be before h.



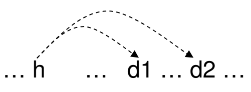


Figure 4: Word pairs and triple for feature representation

in Figure 5, where h is "ate" and d is "with". We can generate the features for the pairs linked by dashed-lines, such as $h - d$, $h - d_{+1}$ and so on. Then we have the temporary bigram-subtrees "ate:1:0-with:2:1" for $h - d$ and "ate:1:0-a:2:1" for $h - d_{+1}$, and so on. If we can find subtree "ate:1:0-with:2:1" for $h - d$ from L_{st} with set ID HF, we generate the feature "H-D:HF", and if we find subtree "ate:1:0-a:2:1" for $h - d_{+1}$ with set ID ZERO, we generate the feature "H-D+1:ZERO". The other three features are also generated similarly.

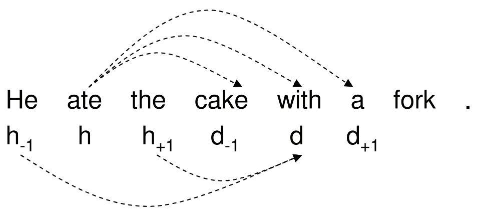


Figure 5: First-order subtree-based features

3.3.2 Second-order subtree-based features

The second-order features are based on trigram-subtrees that are related to triples of words. We generate features for a triple of a head h, its dependent $d1$, and $d1$'s right-leftmost sibling $d2$. The triple is shown in Figure 4-(b). A temporary trigram-subtree is formed by the word forms of h, $d1$, and $d2$. Then we retrieve the subtree in L_{st} to get its set ID. In addition, we consider the triples of "h-NULL"[6], $d1$, and $d2$, which means that we only check the words of sibling nodes without checking the head word.

Then, we generate second-order subtree-based features, consisting of indicator functions for set IDs of the retrieved trigram-subtrees.

[6]h-NULL is a dummy token

We also generate combined features involving the set IDs and part-of-speech tags of heads, and the set IDs and word forms of heads. Specifically, for any feature related to word form, we remove this feature if the word is not one of the Top-N most frequent words in the training data. We used N=1000 for the experiments in this paper. This method can reduce the size of the feature sets.

In this paper, we only used bigram-subtrees and the limited form of trigram-subtrees, though in theory we can use k-gram-subtrees, which are limited in the same way as our trigram subtrees, in (k-1)th-order MST parsing models mentioned in McDonald and Pereira (2006) or use grandparent-type trigram-subtrees in parsing models of Carreras (2007). Although the higher-order MST parsing models will be slow with exact inference, requiring O(n^k) time (McDonald and Pereira, 2006), it might be possible to use higher-order k-gram subtrees with approximated parsing model in the future. Of course, our method can also be easily extended to the labeled dependency case.

4 Experiments

In order to evaluate the effectiveness of the subtree-based features, we conducted experiments on English data and Chinese Data.

For English, we used the Penn Treebank (Marcus et al., 1993) in our experiments and the tool "Penn2Malt"[7] to convert the data into dependency structures using a standard set of head rules (Yamada and Matsumoto, 2003). To match previous work (McDonald et al., 2005; McDonald and Pereira, 2006; Koo et al., 2008), we split the data into a training set (sections 2-21), a development set (Section 22), and a test set (section 23). Following the work of Koo et al. (2008), we used the MXPOST (Ratnaparkhi, 1996) tagger trained on training data to provide part-of-speech tags for the development and the test set, and we used 10-way jackknifing to generate tags for the training set. For the unannotated data, we used the BLLIP corpus (Charniak et al., 2000) that contains about 43 million words of WSJ text.[8] We used the MXPOST tagger trained on training data to assign part-of-speech tags and used the Basic Parser to process the sentences of the BLLIP corpus.

For Chinese, we used the Chinese Treebank

[7]http://w3.msi.vxu.se/˜nivre/research/Penn2Malt.html

[8]We ensured that the text used for extracting subtrees did not include the sentences of the Penn Treebank.

(CTB) version 4.0[9] in the experiments. We also used the "Penn2Malt" tool to convert the data and created a data split: files 1-270 and files 400-931 for training, files 271-300 for testing, and files 301-325 for development. We used gold standard segmentation and part-of-speech tags in the CTB. The data partition and part-of-speech settings were chosen to match previous work (Chen et al., 2008; Yu et al., 2008). For the unannotated data, we used the PFR corpus[10], which has approximately 15 million words whose segmentation and POS tags are given. We used its original segmentation though there are differences in segmentation policy between CTB and this corpus. As for POS tags, we discarded the original POS tags and assigned CTB style POS tags using a TNT-based tagger (Brants, 2000) trained on the training data. We used the Basic Parser to process all the sentences of the PFR corpus.

We measured the parser quality by the unlabeled attachment score (UAS), i.e., the percentage of tokens (excluding all punctuation tokens) with the correct HEAD. And we also evaluated on complete dependency analysis.

4.1 Experimental Results

In our experiments, we used MSTParser, a freely available implementation[11] of the first- and second-order MST parsing models. For baseline systems, we used the first- and second-order basic features, which were the same as the features used by McDonald and Pereira (2006), and we used the default settings of MSTParser throughout the paper: iters=10; training-k=1; decode-type=proj. We implemented our systems based on the MSTParser by incorporating the subtree-based features.

4.1.1 Main results of English data

	English	
	UAS	Complete
Ord1	90.95	37.45
Ord1s	91.76(+0.81)	40.68
Ord2	91.71	42.88
Ord2s	92.51(+0.80)	46.19
Ord2b	92.28(+0.57)	45.44
Ord2t	92.06(+0.35)	42.96

Table 1: Dependency parsing results for English

[9]http://www.cis.upenn.edu/˜chinese/.

[10]http://www.icl.pku.edu.

[11]http://mstparser.sourceforge.net

The results are shown in Table 1, where Ord1/Ord2 refers to a first-/second-order MSTParser with basic features, Ord1s/Ord2s refers to a first-/second-order MSTParser with basic+subtree-based features, and the improvements by the subtree-based features over the basic features are shown in parentheses. Note that we use both the bigram- and trigram- subtrees in Ord2s. The parsers using the subtree-based features consistently outperformed those using the basic features. For the first-order parser, we found that there is an absolute improvement of 0.81 points (UAS) by adding subtree-based features. For the second-order parser, we got an absolute improvement of 0.8 points (UAS) by including subtree-based features. The improvements of parsing with subtree-based features were significant in McNemar's Test ($p < 10^{-6}$).

We also checked the sole effect of bigram- and trigram-subtrees. The results are also shown in Table 1, where Ord2b/Ord2t refers to a second-order MSTParser with bigram-/trigram-subtrees only. The results showed that trigram-subtrees can provide further improvement, although the effect of the bigram-subtrees seemed larger.

4.1.2 Comparative results of English data

Table 2 shows the performance of the systems that were compared, where Y&M2003 refers to the parser of Yamada and Matsumoto (2003), CO2006 refers to the parser of Corston-Oliver et al. (2006), Hall2006 refers to the parser of Hall et al. (2006), Wang2007 refers to the parser of Wang et al. (2007), Z&C 2008 refers to the combination graph-based and transition-based system of Zhang and Clark (2008), KOO08-dep1c/KOO08-dep2c refers to a graph-based system with first-/second-order cluster-based features by Koo et al. (2008), and Carreras2008 refers to the paper of Carreras et al. (2008). The results showed that Ord2s performed better than the first five systems. The second-order system of Koo et al. (2008) performed better than our systems. The reason may be that the MSTParser only uses sibling interactions for second-order, while Koo et al. (2008) uses both sibling and grandparent interactions, and uses cluster-based features. Carreras et al. (2008) reported a very high accuracy using information of constituent structure of the TAG grammar formalism. In our systems, we did not use such knowledge.

Our subtree-based features could be combined with the techniques presented in other work, such as the cluster-based features in Koo et al. (2008), the integrating methods of Zhang and Clark (2008), and Nivre and McDonald (2008), and the parsing methods of Carreras et al. (2008).

	English	
	UAS	Complete
Y&M2003	90.3	38.4
CO2006	90.8	37.6
Hall2006	89.4	36.4
Wang2007	89.2	34.4
Z&C2008	92.1	45.4
KOO08-dep1c	92.23	–
KOO08-dep2c	93.16	–
Carreras2008	93.5	–
Ord1	90.95	37.45
Ord1s	91.76	40.68
Ord1c	91.88	40.71
Ord1i	91.68	41.43
Ord1sc	92.20	42.98
Ord1sci	92.60	44.28
Ord2	91.71	42.88
Ord2s	92.51	46.19
Ord2c	92.40	44.08
Ord2i	92.12	44.37
Ord2sc	92.70	46.56
Ord2sci	93.16	47.15

Table 2: Dependency parsing results for English, for our parsers and previous work

To demonstrate that our approach and other work are complementary, we thus implemented a system using all the techniques we had at hand that used subtree- and cluster-based features and applied the integrating method of Nivre and McDonald (2008). We used the word clustering tool[12], which was used by Koo et al. (2008), to produce word clusters on the BLLIP corpus. The cluster-based features were the same as the features used by Koo et al. (2008). For the integrating method, we used the transition MaxEnt-based parser of Zhao and Kit (2008) because it was faster than the MaltParser. The results are shown in the bottom part of Table 2, where Ord1c/Ord2c refers to a first-/second-order MSTParser with cluster-based features, Ord1i/Ordli refers to a first-/second-order MSTParser with integrating-based features, Ord1sc/Ord2sc refers to a first-/second-order MSTParser with subtree-based+cluster-based features, and Ord1sci/Ord2sci refers to a first-/second-order MSTParser with subtree-based+cluster-based+integrating-based features. Ord1c/Ord2c was worse than KOO08-dep1c/-dep2c, but Ord1sci outperformed KOO08-dep1c and Ord2sci performed similarly to KOO08-dep2c by using all of the techniques we had. These results indicated that subtree-based features can provide different information and work well with other techniques.

[12]http://www.cs.berkeley.edu/˜pliang/software/brown-cluster-1.2.zip

4.1.3 Main results of Chinese data

The results are shown in Table 3 where abbreviations are the same as in Table 1. As in the English experiments, parsers with the subtree-based features outperformed parsers with the basic features, and second-order parsers outperformed first-order parsers. For the first-order parser, the subtree-based features provided 1.3 absolute points improvement. For the second-order parser, the subtree-based features achieved an absolute improvement of 1.25 points. The improvements of parsing with subtree-based features were significant in McNemar's Test ($p < 10^{-5}$).

	Chinese	
	UAS	Complete
Ord1	86.38	40.80
Ord1s	87.68(+1.30)	42.24
Ord2	88.18	47.12
Ord2s	89.43(+1.25)	47.53
Ord2b	89.16(+0.98)	47.12
Ord2t	88.55(+0.37)	47.12

Table 3: Dependency parsing results for Chinese.

4.1.4 Comparative results of Chinese data

Table 4 shows the comparative results, where Wang2007 refers to the parser of Wang et al. (2007), Chen2008 refers to the parser of Chen et al. (2008), and Yu2008 refers to the parser of Yu et al. (2008) that is the best reported results for this data set. And "all words" refers to all the sentences in test set and "$\leq$ 40 words"[13] refers to the sentences with the length up to 40. The table shows that our parsers outperformed previous systems.

We also implemented integrating systems for Chinese data as well. When we applied the cluster-based features, the performance dropped a little. The reason may be that we are using gold-POS tags for Chinese data[14]. Thus we did not

[13]Wang et al. (2007) and Chen et al. (2008) reported the scores on these sentences.

[14]We tried to use the cluster-based features for Chinese with the same setting of POS tags as English data, then the cluster-based features did provide improvement.

use cluster-based features for the integrating systems. The results are shown in Table 4, where Ord1si/Ord2si refers to the first-order/second-order system with subtree-based+intergrating-based features. We found that the integrating systems provided better results. Overall, we have achieved a high accuracy, which is the best known result for this dataset.

Zhang and Clark (2008) and Duan et al. (2007) reported results on a different data split of Penn Chinese Treebank. We also ran our systems (Ord2s) on their data and provided UAS 86.70 (for non-root words)/77.39 (for root words), better than their results: 86.21/76.26 in Zhang and Clark (2008) and 84.36/73.70 in Duan et al. (2007).

	Chinese			
	all words		$\leq$ 40 words	
	UAS	Complete	UAS	Complete
Wang2007	–	–	86.6	28.4
Chen2008	86.52	–	88.4	–
Yu2008	87.26	–	–	–
Ord1s	87.68	42.24	91.11	54.40
Ord1si	88.24	43.96	91.32	55.93
Ord2s	89.43	47.53	91.67	59.77
Ord2si	89.91	48.56	92.34	62.83

Table 4: Dependency parsing results for Chinese, for our parsers and for previous work

4.1.5 Effect of different sizes of unannotated data

Here, we consider the improvement relative to the sizes of the unannotated data. Figure 6 shows the results of first-order parsers with different numbers of words in the unannotated data. Please note that the size of full English unannotated data is 43M and the size of full Chinese unannotated data is 15M. From the figure, we found that the parser obtained more benefits as we added more unannotated data.

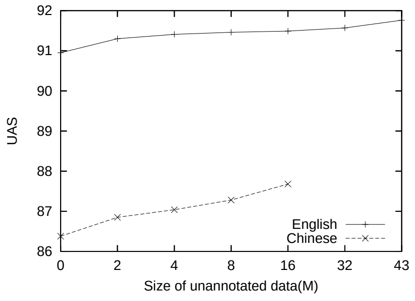


Figure 6: Results with different sizes of large-scale unannotated data.

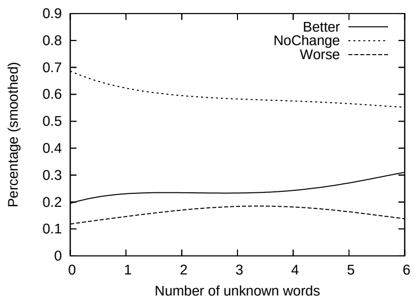


Figure 7: Improvement relative to unknown words for English

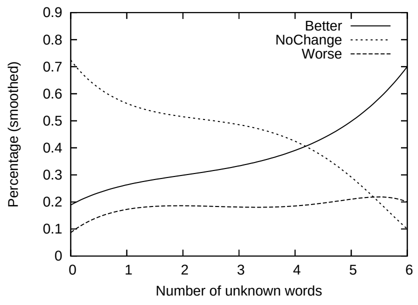


Figure 8: Improvement relative to unknown words for Chinese

4.2 Additional Analysis

In this section, we investigated the results on sentence level from different views. For Figures 7-12, we classified each sentence into one of three classes: "Better" for those where the proposed parsers provided better results relative to the parsers with basic features, "Worse" for those where the proposed parsers provided worse results relative to the basic parsers, and "NoChange" for those where the accuracies remained the same.

4.2.1 Unknown words

Here, we consider the unknown word[15] problem, which is an important issue for parsing. We calculated the number of unknown words in one sentence, and listed the changes of the sentences with unknown words. Here, we compared the Ord1 system and the Ord1s system.

Figures 7 and 8 show the results, where the x axis refers to the number of unknown words in one sentence and the y axis shows the percentages of the three classes. For example, for the sentences having three unknown words in the Chinese data, 31.58% improved, 23.68% worsened, and 44.74% were unchanged. We did not show the results of

[15]An unknown word is a word that is not included in the training data.

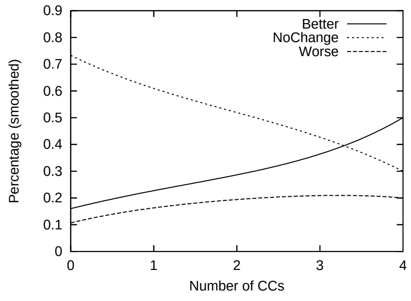


Figure 9: Improvement relative to number of conjunctions for English

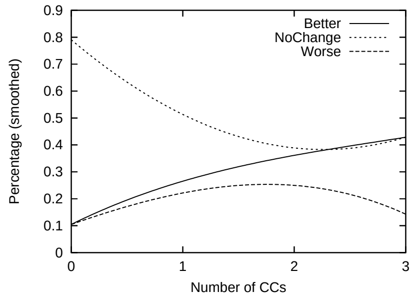


Figure 10: Improvement relative to number of conjunctions for Chinese

the sentences with more than six unknown words because their numbers were very small. The Better and Worse curves showed that our approach always provided better results. The results indicated that the improvements apparently became larger when the sentences had more unknown words for the Chinese data. And for the English data, the graph also showed the similar trend, although the improvements for the sentences have three and four unknown words were slightly less than the others.

4.2.2 Coordinating conjunctions

We analyzed our new parsers' behavior for coordinating conjunction structures, which is a very difficult problem for parsing (Kawahara and Kurohashi, 2008). Here, we compared the Ord2 system with the Ord2s system.

Figures 9 and 10 show how the subtree-based features affect accuracy as a function of the number of conjunctions, where the x axis refers to the number of conjunctions in one sentence and the y axis shows the percentages of the three classes. The figures indicated that the subtree-based features improved the coordinating conjunction problem. In the trigram-subtree list, many subtrees are related to coordinating conjunctions, such as "utilities:1:3 and:2:3 businesses:3:0" and "pull:1:0 and:2:1 protect:3:1". These subtrees can provide additional information for parsing models.

4.2.3 PP attachment

We analyzed our new parsers' behavior for preposition-phrase attachment, which is also a difficult task for parsing (Ratnaparkhi et al., 1994). We compared the Ord2 system with the Ord2s system. Figures 11 and 12 show how the subtree-based features affect accuracy as a function of the number of prepositions, where the x axis refers to the number of prepositions in one sentence and the y axis shows the percentages of the three classes. The figures indicated that the subtree-based features improved preposition-phrase attachment.

5 Related work

Our approach is to incorporate unannotated data into parsing models for dependency parsing. Several previous studies relevant to our approach have been conducted.

Chen et al. (2008) previously proposed an approach that used the information on short dependency relations for Chinese dependency parsing. They only used the word pairs within two word distances for a transition-based parsing algorithm. The approach in this paper differs in that we use richer information on trigram-subtrees besides bigram-subtrees that contain word pairs. And our work is focused on graph-based parsing models as opposed to transition-based models. Yu et al. (2008) constructed case structures from auto-parsed data and utilized them in parsing. Compared with their method, our method is much simpler but has great effects.

Koo et al. (2008) used the Brown algorithm to produce word clusters on large-scale unannotated data and represented new features based on the clusters for parsing models. The cluster-based features provided very impressive results. In addition, they used the parsing model by Carreras (2007) that applied second-order features on both sibling and grandparent interactions. Note that our approach and their approach are complementary in that we can use both subtree- and cluster-based features for parsing models. The experimental results showed that we achieved better accuracy for first-order models when we used both of these two types of features.

Sagae and Tsujii (2007) presented an co-training approach for dependency parsing adap-

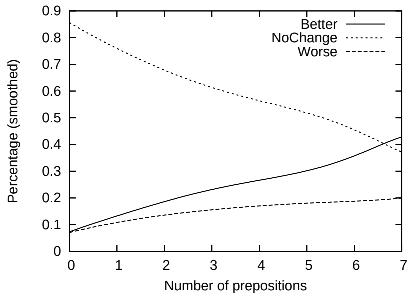


Figure 11: Improvement relative to number of prepositions for English

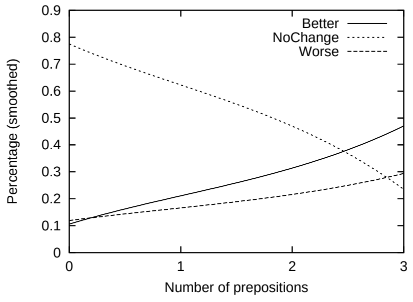


Figure 12: Improvement relative to number of prepositions for Chinese

tation. They used two parsers to parse the sentences in unannotated data and selected only identical results produced by the two parsers. Then, they retrained a parser on newly parsed sentences and the original labeled data. Our approach represents subtree-based features on the original gold-standard data to retrain parsers. McClosky et al. (2006) presented a self-training approach for phrase structure parsing and the approach was shown to be effective in practice. However, their approach depends on a high-quality reranker, while we simply augment the features of an existing parser. Moreover, we could use the output of our systems for co-training/self-training techniques.

6 Conclusions

We present a simple and effective approach to improve dependency parsing using subtrees from auto-parsed data. In our method, first we use a baseline parser to parse large-scale unannotated data, and then we extract subtrees from dependency parsing trees in the auto-parsed data. Finally, we construct new subtree-based features for parsing models. The results show that our approach significantly outperforms baseline systems. We also show that our approach and other techniques are complementary, and then achieve the best reported accuracy for the Chinese data and an accuracy that is competitive with the best known systems for the English data.

References

T. Brants. 2000. TnT–a statistical part-of-speech tagger. *Proceedings of ANLP*, pages 224–231.

S. Buchholz, E. Marsi, A. Dubey, and Y. Krymolowski. 2006. CoNLL-X shared task on multilingual dependency parsing. *Proceedings of CoNLL-X*.

Xavier Carreras, Michael Collins, and Terry Koo. 2008. Tag, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *Proceedings of CoNLL 2008*, pages 9–16, Manchester, England, August. Coling 2008 Organizing Committee.

X. Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 957–961.

E. Charniak, D. Blaheta, N. Ge, K. Hall, J. Hale, and M. Johnson. 2000. BLLIP 1987-89 WSJ Corpus Release 1, LDC2000T43. *Linguistic Data Consortium.*

WL. Chen, D. Kawahara, K. Uchimoto, YJ. Zhang, and H. Isahara. 2008. Dependency parsing with short dependency relations in unlabeled data. In *Proceedings of IJCNLP 2008.*

S. Corston-Oliver, A. Aue, Kevin. Duh, and Eric Ringger. 2006. Multilingual dependency parsing using bayes point machines. In *HLT-NAACL2006.*

H. Cui, RX. Sun, KY. Li, MY. Kan, and TS. Chua. 2005. Question answering passage retrieval using dependency relations. In *Proceedings of SIGIR 2005*, pages 400–407, New York, NY, USA. ACM.

Xiangyu Duan, Jun Zhao, and Bo Xu. 2007. Probabilistic models for action-based chinese dependency parsing. In *Proceedings of ECML/ECPPKDD*, Warsaw, Poland.

Johan Hall, Joakim Nivre, and Jens Nilsson. 2006. Discriminative classifiers for deterministic dependency parsing. In *In Proceedings of CoLING-ACL.*

D. Kawahara and S. Kurohashi. 2008. Coordination disambiguation without any similarities. In *Proceedings of Coling 2008*, pages 425–432, Manchester, UK, August.

T. Koo, X. Carreras, and M. Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL-08: HLT*, Columbus, Ohio, June.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguisticss*, 19(2):313–330.

D. McClosky, E. Charniak, and M. Johnson. 2006. Reranking and self-training for parser adaptation. In *Proceedings of Coling-ACL*, pages 337–344.

R. McDonald and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proc. of EACL2006*.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proc. of ACL 2005*.

T. Nakazawa, K. Yu, D. Kawahara, and S. Kurohashi. 2006. Example-based machine translation based on deeper nlp. In *Proceedings of IWSLT 2006*, pages 64–70, Kyoto, Japan.

J. Nivre and R. McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL-08: HLT*, Columbus, Ohio, June.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932.

A. Ratnaparkhi, J. Reynar, and S. Roukos. 1994. A maximum entropy model for prepositional phrase attachment. In *Proceedings of HLT*, pages 250–255.

A. Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of EMNLP*, pages 133–142.

K. Sagae and J. Tsujii. 2007. Dependency parsing and domain adaptation with LR models and parser ensembles. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 1044–1050.

M. Steedman, M. Osborne, A. Sarkar, S. Clark, R. Hwa, J. Hockenmaier, P. Ruhlen, S. Baker, and J. Crim. 2003. Bootstrapping statistical parsers from small datasets. In *Proceedings of EACL 2003*, pages 331–338.

Gertjan van Noord. 2007. Using self-trained bilexical preferences to improve disambiguation accuracy. In *Proceedings of IWPT-07*, June.

Qin Iris Wang, Dekang Lin, and Dale Schuurmans. 2007. Simple training of dependency parsers via structured boosting. In *Proceedings of IJCAI2007*.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT2003*, pages 195–206.

K. Yu, D. Kawahara, and S. Kurohashi. 2008. Chinese dependency parsing with large scale automatically constructed case structures. In *Proceedings of Coling 2008*, pages 1049–1056, Manchester, UK, August.

Y. Zhang and S. Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of EMNLP 2008*, pages 562–571, Honolulu, Hawaii, October.

H. Zhao and CY. Kit. 2008. Parsing syntactic and semantic dependencies with two single-stage maximum entropy models. In *Proceedings of CoNLL 2008*, pages 203–207, Manchester, England, August.