Minimally Lexicalized Dependency Parsing

Daisuke Kawahara and **Kiyotaka Uchimoto**
National Institute of Information and Communications Technology,
3-5 Hikaridai Seika-cho Soraku-gun, Kyoto, 619-0289, Japan
{dk, uchimoto}@nict.go.jp

Abstract

Dependency structures do not have the information of phrase categories in phrase structure grammar. Thus, dependency parsing relies heavily on the lexical information of words. This paper discusses our investigation into the effectiveness of lexicalization in dependency parsing. Specifically, by restricting the degree of lexicalization in the training phase of a parser, we examine the change in the accuracy of dependency relations. Experimental results indicate that minimal or low lexicalization is sufficient for parsing accuracy.

1 Introduction

In recent years, many accurate phrase-structure parsers have been developed (e.g., (Collins, 1999; Charniak, 2000)). Since one of the characteristics of these parsers is the use of lexical information in the tagged corpus, they are called "lexicalized parsers". Unlexicalized parsers, on the other hand, achieved accuracies almost equivalent to those of lexicalized parsers (Klein and Manning, 2003; Matsuzaki et al., 2005; Petrov et al., 2006). Accordingly, we can say that the state-of-the-art lexicalized parsers are mainly based on unlexical (grammatical) information due to the sparse data problem. Bikel also indicated that Collins' parser can use bilexical dependencies only 1.49% of the time; the rest of the time, it backs off to condition one word on just phrasal and part-of-speech categories (Bikel, 2004).

This paper describes our investigation into the effectiveness of lexicalization in dependency parsing instead of phrase-structure parsing. Usual dependency parsing cannot utilize phrase categories, and thus relies on word information like parts of speech and lexicalized words. Therefore, we want to know the performance of dependency parsers that have minimal or low lexicalization.

Dependency trees have been used in a variety of NLP applications, such as relation extraction (Culotta and Sorensen, 2004) and machine translation (Ding and Palmer, 2005). For such applications, a fast, efficient and accurate dependency parser is required to obtain dependency trees from a large corpus. From this point of view, minimally lexicalized parsers have advantages over fully lexicalized ones in parsing speed and memory consumption.

We examined the change in performance of dependency parsing by varying the degree of lexicalization. The degree of lexicalization is specified by giving a list of words to be lexicalized, which appear in a training corpus. For minimal lexicalization, we used a short list that consists of only high-frequency words, and for maximal lexicalization, the whole list was used. Consequently, minimally or low lexicalization is sufficient for dependency accuracy.

2 Related Work

Klein and Manning presented an unlexicalized PCFG parser that eliminated all the lexicalized parameters (Klein and Manning, 2003). They manually split category tags from a linguistic view. This corresponds to determining the degree of lexicalization by hand. Their parser achieved an F_1 of 85.7% for section 23 of the Penn Treebank. Matsuzaki et al. and Petrov et al. proposed an automatic approach to

Proceedings of the ACL 2007 Demo and Poster Sessions, pages 205–208,
Prague, June 2007. ©2007 Association for Computational Linguistics

Dependency accuracy (DA) Proportions of words, except punctuation marks, that are assigned the correct heads.

Root accuracy (RA) Proportions of root words that are correctly detected.

Complete rate (CR) Proportions of sentences whose dependency structures are completely correct.

Table 1: Evaluation criteria.

	DA	RA	CR
(Yamada and Matsumoto, 2003)	90.3	91.6	38.4
(Nivre and Scholz, 2004)	87.3	84.3	30.4
(Isozaki et al., 2004)	91.2	95.7	40.7
(McDonald et al., 2005)	90.9	94.2	37.5
(McDonald and Pereira, 2006)	91.5	N/A	42.1
(Corston-Oliver et al., 2006)	90.8	93.7	37.6
Our Base Parser	90.9	92.6	39.2

Table 2: Comparison of parser performance.

splitting tags (Matsuzaki et al., 2005; Petrov et al., 2006). In particular, Petrov et al. reported an F_1 of 90.2%, which is equivalent to that of state-of-the-art lexicalized parsers.

Dependency parsing has been actively studied in recent years (Yamada and Matsumoto, 2003; Nivre and Scholz, 2004; Isozaki et al., 2004; McDonald et al., 2005; McDonald and Pereira, 2006; Corston-Oliver et al., 2006). For instance, Nivre and Scholz presented a deterministic dependency parser trained by memory-based learning (Nivre and Scholz, 2004). McDonald et al. proposed an online large-margin method for training dependency parsers (McDonald et al., 2005). All of them performed experiments using section 23 of the Penn Treebank. Table 2 summarizes their dependency accuracies based on three evaluation criteria shown in Table 1. These parsers believed in the generalization ability of machine learners and did not pay attention to the issue of lexicalization.

3 Minimally Lexicalized Dependency Parsing

We present a simple method for changing the degree of lexicalization in dependency parsing. This method restricts the use of lexicalized words, so it is the opposite to tag splitting in phrase-structure parsing. In the remainder of this section, we first describe a base dependency parser and then report experimental results.

3.1 Base Dependency Parser

We built a parser based on the deterministic algorithm of Nivre and Scholz (Nivre and Scholz, 2004) as a base dependency parser. We adopted this algorithm because of its linear-time complexity.

In the algorithm, parsing states are represented by triples $\langle S, I, A\rangle$, where S is the stack that keeps the words being under consideration, I is the list of remaining input words, and A is the list of determined dependencies. Given an input word sequence, W, the parser is first initialized to the triple $\langle nil, W, \phi\rangle$[1]. The parser estimates a dependency relation between two words (the top elements of stacks S and I). The algorithm iterates until the list I is empty. There are four possible operations for a parsing state (where t is the word on top of S, n is the next input word in I, and w is any word):

Left In a state $\langle t|S, n|I, A\rangle$, if there is no dependency relation $(t \rightarrow w)$ in A, add the new dependency relation $(t \rightarrow n)$ into A and pop S (remove t), giving the state $\langle S, n|I, A \cup (t \rightarrow n)\rangle$.

Right In a state $\langle t|S, n|I, A\rangle$, if there is no dependency relation $(n \rightarrow w)$ in A, add the new dependency relation $(n \rightarrow t)$ into A and push n onto S, giving the state $\langle n|t|S, I, A \cup (n \rightarrow t)\rangle$.

Reduce In a state $\langle t|S, I, A\rangle$, if there is a dependency relation $(t \rightarrow w)$ in A, pop S, giving the state $\langle S, I, A\rangle$.

Shift In a state $\langle S, n|I, A\rangle$, push n onto S, giving the state $\langle n|S, I, A\rangle$.

In this work, we used Support Vector Machines (SVMs) to predict the operation given a parsing state. Since SVMs are binary classifiers, we used the pair-wise method to extend them in order to classify our four-class task.

The features of a node are the word's lemma, the POS/chunk tag and the information of its child node(s). The lemma is obtained from the word form using a lemmatizer, except for numbers, which are replaced by "$\langle$num$\rangle$". The context features are the two preceding nodes of node t (and t itself), the two succeeding nodes of node n (and n itself), and their

[1]We use "nil" to denote an empty list and $a|A$ to denote a list with head a and tail A.

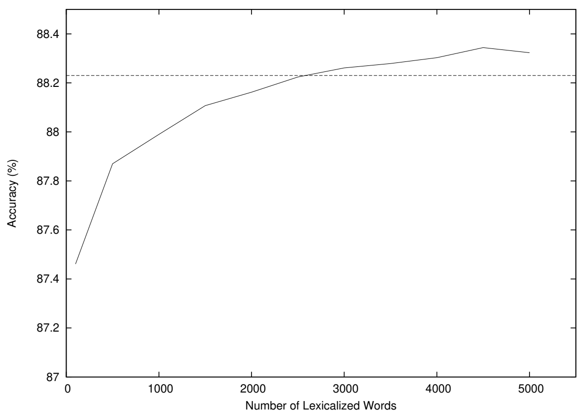

Figure 1: Dependency accuracies on the WSJ while changing the degree of lexicalization.

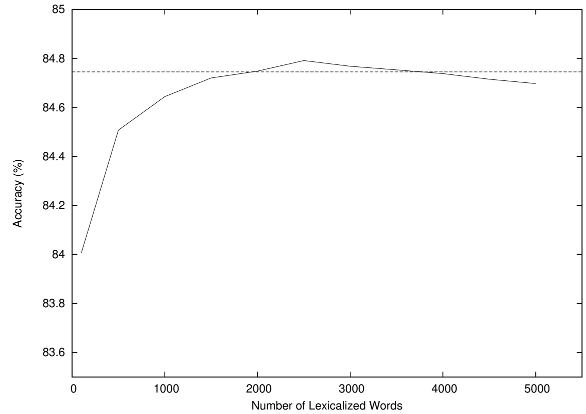

Figure 2: Dependency accuracies on the Brown Corpus while changing the degree of lexicalization.

child nodes (lemmas and POS tags). The distance between nodes n and t is also used as a feature.

We trained our models on sections 2-21 of the WSJ portion of the Penn Treebank. We used section 23 as the test set. Since the original treebank is based on phrase structure, we converted the treebank to dependencies using the head rules provided by Yamada [2]. During the training phase, we used intact POS and chunk tags[3]. During the testing phase, we used automatically assigned POS and chunk tags by Tsuruoka's tagger[4](Tsuruoka and Tsujii, 2005) and YamCha chunker[5](Kudo and Matsumoto, 2001). We used an SVMs package, TinySVM[6],and trained the SVMs classifiers using a third-order polynomial kernel. The other parameters are set to default.

The last row in Table 2 shows the accuracies of our base dependency parser.

3.2 Degree of Lexicalization vs. Performance

The degree of lexicalization is specified by giving a list of words to be lexicalized, which appear in a training corpus. For minimal lexicalization, we used a short list that consists of only high-frequency words, and for maximal lexicalization, the whole list was used.

To conduct the experiments efficiently, we trained our models using the first 10,000 sentences in sections 2-21 of the WSJ portion of the Penn Treebank. We used section 24, which is usually used as the development set, to measure the change in performance based on the degree of lexicalization.

We counted word (lemma) frequencies in the training corpus and made a word list in descending order of their frequencies. The resultant list consists of 13,729 words, and the most frequent word is "the", which occurs 13,252 times, as shown in Table 3. We define the degree of lexicalization as a threshold of the word list. If, for example, this threshold is set to 1,000, the top 1,000 most frequently occurring words are lexicalized.

We evaluated dependency accuracies while changing the threshold of lexicalization. Figure 1 shows the result. The dotted line (88.23%) represents the dependency accuracy of the maximal lexicalization, that is, using the whole word list. We can see that the decrease in accuracy is less than 1% at the minimal lexicalization (degree=100) and the accuracy of more than 3,000 degree slightly exceeds that of the maximal lexicalization. The best accuracy (88.34%) was achieved at 4,500 degree and significantly outperformed the accuracy (88.23%) of the maximal lexicalization (McNemar's test; $p = 0.017 < 0.05$). These results indicate that maximal lexicalization is not so effective for obtaining accurate dependency relations.

We also applied the same trained models to the Brown Corpus as an experiment of parser adaptation. We first split the Brown Corpus portion of

[2]http://www.jaist.ac.jp/˜h-yamada/

[3]In a preliminary experiment, we tried to use automatically assigned POS and chunk tags, but we did not detect significant difference in performance.

[4]http://www-tsujii.is.s.u-tokyo.ac.jp/˜tsuruoka/postagger/

[5]http://chasen.org/˜taku-ku/software/yamcha/

[6]http://chasen.org/˜taku-ku/software/TinySVM/

rank	word	freq.	rank	word	freq.
1	the	13,252	1,000	watch	29
2	,	12,858	⋮	⋮	⋮
⋮	⋮	⋮	2,000	healthvest	12
100	week	261	⋮	⋮	⋮
⋮	⋮	⋮	3,000	whoop	7
500	estate	64	⋮	⋮	⋮
⋮	⋮	⋮			

Table 3: Word list.

the Penn Treebank into training and testing parts in the same way as (Roark and Bacchiani, 2003). We further extracted 2,425 sentences at regular intervals from the training part and used them to measure the change in performance while varying the degree of lexicalization. Figure 2 shows the result. The dotted line (84.75%) represents the accuracy of maximal lexicalization. The resultant curve is similar to that of the WSJ experiment[7]. We can say that our claim is true even if the testing corpus is outside the domain.

3.3 Discussion

We have presented a minimally or lowly lexicalized dependency parser. Its dependency accuracy is close or almost equivalent to that of fully lexicalized parsers, despite the lexicalization restriction. Furthermore, the restriction reduces the time and space complexity. The minimally lexicalized parser (degree=100) took 12m46s to parse the WSJ development set and required 111 MB memory. These are 36% of time and 45% of memory reduction, compared to the fully lexicalized one.

The experimental results imply that training corpora are too small to demonstrate the full potential of lexicalization. We should consider unsupervised or semi-supervised ways to make lexicalized parsers more effective and accurate.

Acknowledgment

This research is partially supported by special coordination funds for promoting science and technology.

[7] In the experiment on the Brown Corpus, the difference between the best accuracy and the baseline was not significant.

References

Daniel M. Bikel. 2004. Intricacies of Collins' parsing model. *Computational Linguistics*, 30(4):479–511.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL2000*, pages 132–139.

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

Simon Corston-Oliver, Anthony Aue, Kevin Duh, and Eric Ringger. 2006. Multilingual dependency parsing using bayes point machines. In *Proceedings of HLT-NAACL2006*, pages 160–167.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of ACL2004*, pages 423–429.

Yuan Ding and Martha Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proceedings of ACL2005*, pages 541–548.

Hideki Isozaki, Hideto Kazawa, and Tsutomu Hirao. 2004. A deterministic word dependency analyzer enhanced with preference learning. In *Proceedings of COLING2004*, pages 275–281.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL2003*, pages 423–430.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *Proceedings of NAACL2001*, pages 192–199.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings of ACL2005*, pages 75–82.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL2006*, pages 81–88.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL2005*, pages 91–98.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of English text. In *Proceedings of COLING2004*, pages 64–70.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of COLING-ACL2006*, pages 433–440.

Brian Roark and Michiel Bacchiani. 2003. Supervised and unsupervised PCFG adaptation to novel domains. In *Proceedings of HLT-NAACL2003*, pages 205–212.

Yoshimasa Tsuruoka and Jun'ichi Tsujii. 2005. Bidirectional inference with the easiest-first strategy for tagging sequence data. In *Proceedings of HLT-EMNLP2005*, pages 467–474.

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