Training Dependency Parsers from Partially Annotated Corpora

Daniel Flannery[1], **Yusuke Miyao**[2], **Graham Neubig**[1], **Shinsuke Mori**[1]

[1]Graduate School of Informatics, Kyoto University
Yoshida Honmachi, Sakyo-ku, Kyoto, Japan

[2]National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, Japan

flannery@ar.media.kyoto-u.ac.jp, yusuke@nii.ac.jp,
neubig@ar.media.kyoto-u.ac.jp, forest@i.kyoto-u.ac.jp

Abstract

We introduce a maximum spanning tree (MST) dependency parser that can be trained from partially annotated corpora, allowing for effective use of available linguistic resources and reduction of the costs of preparing new training data. This is especially important for domain adaptation in a real-world situation. We use a pointwise approach where each edge in the dependency tree for a sentence is estimated independently. Experiments on Japanese dependency parsing show that this approach allows for rapid training and achieves accuracy comparable to state-of-the-art dependency parsers trained on fully annotated data.

1 Introduction

Parsing is one of the fundamental building blocks of natural language processing, with applications ranging from machine translation (Yamada and Knight, 2001) to information extraction (Miyao et al., 2009). However, while statistical parsers achieve higher and higher accuracies on in-domain text, the creation of data to train these parsers is labor-intensive, which becomes a bottleneck for smaller languages. In addition, it is also a well known fact that accuracy plummets when tested on sentences of a different domain than the training corpus (Gildea, 2001; Petrov et al., 2010), and that in-domain data can be annotated to make up for this weakness.

In this paper, we propose a maximum spanning tree (MST) parser that helps ameliorate these problems by allowing for the efficient development of training data. This is done through a combination of an efficient corpus annotation strategy, and a novel parsing method. For corpus construction, we use partial annotation, which allows an annotator to skip annotation of unnecessary edges, focusing their efforts only on the ones that will provide the maximal gains in accuracy.

While partial annotation has been shown to be an effective annotation strategy for a number of tasks (Tsuboi et al., 2008; Sassano and Kurohashi, 2010; Neubig and Mori, 2010), traditional MST parsers such as that of McDonald et al. (2005) cannot be learned from partially annotated data. The reason for this is that they use structural prediction methods that must be learned from fully annotated sentences. However, a number of recent works (Liang et al., 2008; Neubig et al., 2011) have found that it is possible to ignore structure and still achieve competitive accuracy on tasks such as part-of-speech (POS) tagging.

Similarly, recent work on dependency parsing (Spreyer and Kuhn, 2009; Spreyer et al., 2010) has shown that training constraints can be relaxed to allow MST parsers to be trained from partially annotated sentences, with only a small reduction in parsing accuracy. In this approach the scoring function used to evaluate potential dependency trees is modified so that it does not penalize trees consistent with the partial annotations used for training. Our formulation of an MST parser is based on an even stronger independence assumption, namely that the score of each edge is independent of the other edges in the dependency tree. While this does have the potential to decrease accuracy, it has a number of advantages such as the ability to use partially annotated data, faster speed, and simple implementation.

We perform an evaluation of the proposed method on a Japanese dependency parsing task. First, we compare the proposed method to both a traditional MST parser (McDonald et al., 2005), and a deterministic parser (Nivre and Scholz, 2004). We find that despite the lack of structure in our prediction method, the proposed method is still able to achieve accuracy similar to that of the traditional MST parser, while training and testing speeds are similar to that of the deterministic parser.

In addition, we perform a case-study of the use of partial annotation in a practical scenario, where we have data that follows a segmentation standard that differs from the one we would like to follow. In Japanese dependency parsing, traditionally phrase segments (*bunsetsu*) have been used instead of words as the minimal unit for parsing (Kudo and Matsumoto, 2002; Sassano and Kurohashi, 2010), but these segments are often too large or unwieldy for applications such as information extraction and machine translation (Nakazawa and Kurohashi, 2008). In our case-study, we demonstrate that a corpus labeled with phrase dependencies can be used as a partially annotated corpus in the development of a word-based parser that is more appropriate for these applications. The use of a phrase-labeled corpus allows us to increase the accuracy of a word-based parser trained on a smaller word-labeled data set by 2.75%.

2 Pointwise estimation for dependency parsing

This work follows the standard setting of recent work on dependency parsing (Buchholz and Marsi, 2006). Given as input a sequence of words, $\boldsymbol{w} = \langle w_1, w_2, \ldots, w_n \rangle$, the goal is to output a dependency tree $\boldsymbol{d} = \langle d_1, d_2, \ldots, d_n \rangle$, where $d_i \equiv j$ when the head of w_i is w_j.[1] We assume that $d_i = 0$ for some word w_i in a sentence, which indicates that w_i is the head of the sentence.

2.1 A Pointwise MST Parser

The parsing model we pursue in this paper is McDonald et al. (2005)'s edge-factored model. A score, $\sigma(d_i)$, is assigned to each edge (i.e. dependency) d_i, and parsing finds a dependency tree, $\hat{\boldsymbol{d}}$, that maximizes the sum of the scores of all the edges.

$$\hat{\boldsymbol{d}} = \underset{\boldsymbol{d}}{\mathbf{argmax}} \sum_{d \in \boldsymbol{d}} \sigma(d). \tag{1}$$

It is known that, given $\sigma(d)$ for all possible dependencies in a sentence, $\hat{\boldsymbol{d}}$ can be computed by the maximum spanning tree algorithm such as Chu-Liu/Edmonds' algorithm.

An important difference from McDonald et al. (2005) is in the estimation of $\sigma(d)$. McDonald et al. (2005) applied a perceptron-like algorithm that optimizes the score of entire dependency trees. However, we stick to pointwise prediction: $\sigma(d_i)$ is estimated for each w_i independently. A variety of machine-learning-based classifiers can be applied to the estimation of $\sigma(d)$, because it is essentially an n-class classification problem.

In the experiments, we estimate a log-linear model (Berger et al., 1996). We calculate the probability of a dependency labeling $p(d_i = j)$ for a word w_i from its context, which is a tuple $x = \langle \boldsymbol{w}, \boldsymbol{t}, i \rangle$, where $\boldsymbol{t} = \langle t_1, t_2, \ldots, t_n \rangle$ is a sequence of POS tags assigned to $\boldsymbol{w}$ by a tagger. The conditional probability $p(j|x)$ is given by the following equation.

$$p(j|x, \boldsymbol{\theta}) = \frac{\exp\left(\boldsymbol{\theta} \cdot \boldsymbol{\phi}(x, j)\right)}{\sum_{j' \in \mathcal{J}} \exp\left(\boldsymbol{\theta} \cdot \boldsymbol{\phi}(x, j')\right)} \tag{2}$$

The feature vector $\boldsymbol{\phi} = \langle \phi_1, \phi_2, \ldots, \phi_m \rangle$ is a vector of non-negative values calculated from features on pairs (x, j), with corresponding weights given by the parameter vector $\boldsymbol{\theta} = \langle \theta_1, \theta_2, \ldots, \theta_m \rangle$. We estimate $\boldsymbol{\theta}$ from sentences annotated with dependencies. It should be noted that the probability $p(d_i)$ depends only on i, j, and the inputs $\boldsymbol{w}$, $\boldsymbol{t}$, which ensures that it is estimated independently for each w_i. Because parameter estimation does not involve computing $\hat{\boldsymbol{d}}$, we do not apply the maximum spanning tree algorithm in training.

2.2 Features

Our current implementation uses the following features for $\boldsymbol{\phi}$.

F1: The distance between a dependent word and its candidate head.

F2: The surface forms of the dependent and head words.

F3: The parts-of-speech of the dependent and head words.

F4: The surface forms of up to three words to the left of the dependent and head words.

F5: The surface forms of up to three words to the right of the dependent and head words.

[1] While we describe unlabeled dependency parsing for simplicity, it is trivial to extend it to labeled dependency parsing.

i	1	2	3	4	5	6	7	8	9
w_i	政府	は	投資	に	つなが	る	と	歓迎	し
Eng.	Gov.	*subj.*	investment	to	leads	*ending*	that	welcomes	do
t_i	noun	part.	noun	part.	verb	infl.	part.	noun	verb
d_i		8							
F1		6							
F2		は						歓迎	
F3		part.						noun	
F4		政府						つなが, る, と	
F5		投資, に, つなが						し	
F6		noun						verb, infl., part.	
F7		noun, part., verb						verb	

The second word, case marker は (*subj.*), has two grammatically possible heads: the verb つなが (leads) and the verb 歓迎 (welcomes). In our framework, only this word needs to be annotated with its head.

Figure 1: An example of a partially annotated sentence and the features for a dependency between case marker は (*subj.*) and the verb 歓迎 (welcomes).

F6: The parts-of-speech of the words selected for F4.

F7: The parts-of-speech of the words selected for F5.

Figure 1 shows the values of these features for a partially annotated example sentence where one word, case marker は (*subj.*), has been annotated with its head, the verb 歓迎 (welcomes).

Pointwise prediction rather than structured prediction has the potential to hurt parsing accuracy. However, our method can enjoy greater flexibility, which allows for training from partially annotated corpora as will be described in Section 3. It also simplifies the implementation and reduces the time necessary for training.

2.3 Solution Search

In the experiments, we target Japanese parsing. Because Japanese is a head-final language, we assume $d_i > i$ for all $i \neq n$ and $d_n = 0$. This assumption reduces the maximum spanning tree algorithm to a simpler algorithm: for each word we select the dependency with the maximum score. As this never creates a loop of dependencies, a recursive process as in Chu-Liu/Edmonds' algorithm is not necessary.

3 Domain Adaptation for MST Parsing

Assuming that the cost of annotation corresponds roughly to the number of annotations performed, out of all possible annotations to have annotators perform for a target domain corpus we want to select the ones which provide the greatest benefit to accuracy when training. The high cost of annotation work is the primary motivation for this approach.

3.1 Partial Annotation for a Parser

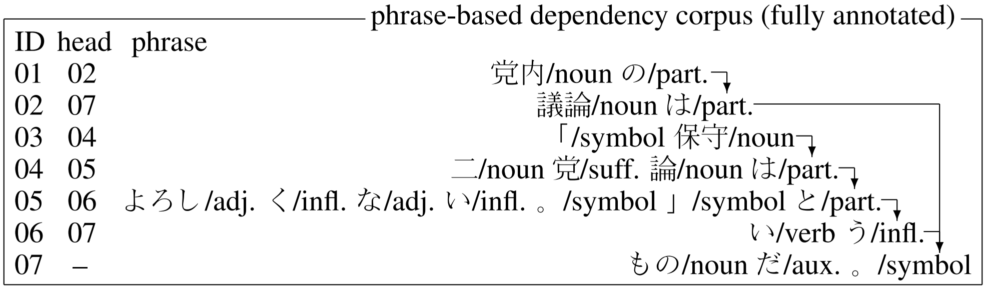
In the context of dependency parsing, partial annotation refers to annotating only certain dependencies between words in a sentence. Dependencies which are assumed to have little to no value for training are left unannotated. Figure 1 shows an example of a partially annotated sentence that can be used as training data by our system.

Before text can be annotated with dependencies for use in our system, it must first be tokenized and labeled with POS tags.[2] We assume that the results of this tokenization and POS tagging are accurate enough that we need to manually annotate only the dependencies between the tokenized words.

3.2 Learning Feature Weights from Partial Annotations

As explained in Section 2.1, edge scores, $\sigma(d_i)$, are estimated for each w_i independently. This means that the estimation of $\sigma(d_i)$ requires only a gold dependency of w_i, and the other dependencies in a sentence are not necessary. This allows us to learn weights $\boldsymbol{\theta}$ for features from partially annotated corpora. When training data includes a gold

[2] We take a language-independent approach that does not make any assumptions about the unit of tokenization or the meaning of tags used.



phrase-based dependency corpus (fully annotated)

ID	head	phrase
01	02	党内/noun の/part.
02	07	議論/noun は/part.
03	04	「/symbol 保守/noun
04	05	二/noun 党/suff. 論/noun は/part.
05	06	よろし/adj. く/infl. な/adj. い/infl. 。/symbol 」/symbol と/part.
06	07	い/verb う/infl.
07	–	もの/noun だ/aux. 。/symbol

Figure 2: An example of phrase-based dependency annotation for a sentence.

dependency that w_i depends on w_j, a discriminative classifier can be trained by regarding $d_i = j$ as a positive sample and $d_i = j'$ s.t. $j' \neq j$ as negative samples.

In the case of Japanese parsing, because $j > i$ for all $d_i = j$, negative samples are $d_i = j'$ s.t. $j' \neq j \wedge j' > i$. For example, from the partial annotation given in Figure 1, we can create a training instance for w_2, は (*subj.*), where the positive sample is $d_2 = 8$ and the negative samples are $d_2 = 3, 4, \ldots, 7, 9$.

3.3 Domain Adaptation with a Partially Annotated Training Corpus

As a case study, we show how partial annotation can be used as a low-cost method of converting the annotation standard of an existing linguistic resource. As we mentioned in Section 1, traditional frameworks for Japanese dependency parsing are phrase-based. Many existing dependency corpora use phrases as the unit of annotation, and these resources are a valuable potential source of data for mining word dependencies. However, phrase dependencies alone do not provide enough information for an automatic conversion to word dependencies. One of the advantages of our parser is that it can be trained on a partially annotated corpus, so if we can derive even some word dependencies from phrase dependencies we can quickly and easily make use of existing resources.

To take advantage of these linguistic resources, we created a number of rules to derive word-based dependency annotations from phrase-based annotations. Instead of trying to convert all phrase dependencies, we focused on heuristics that provide only reliable word dependencies. The word-based dependency set produced by these rules is a partial annotation of the original corpus.

For the domain adaptation experiments described in Section 4, we used this procedure on the NAIST Text Corpus (NTC) (Iida et al., 2007) to create a small partially-annotated target domain corpus. The NTC consists of newspaper articles from the Mainichi Shimbun.[3] Figure 2 shows an example sentence from this corpus annotated with phrase dependencies.

To aid the construction of conversion rules, we chose three broad categories of words - content words, function words, and punctuation symbols - that provide clues to the structure of a phrase. Before we explain our rules, we will give a short explanation of these three categories.

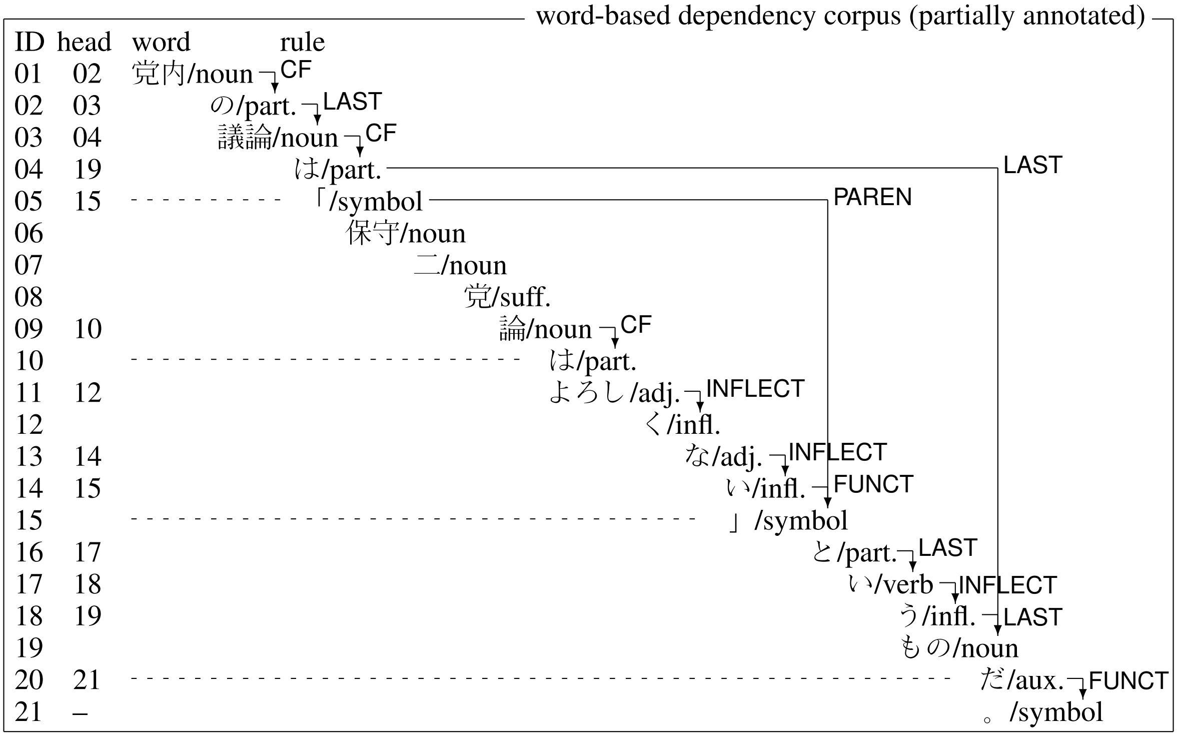
We defined content words as nouns, verbs, adjectives, interjections, prenominal adjectives, suffixes, and prefixes. Function words are auxiliary verbs, particles, inflections, and conjunctions. In this context, punctuation symbols are both the English and Japanese versions of period and comma characters. These three categories are used to determine phrases which can be mined for relatively accurate word dependencies.

Figure 3 shows an example of how the rules explained below are used to derive word-based dependencies from phrase-based dependencies for the sentence given in Figure 2.

The first two rules are inter-phrase rules, which are concerned with the relationship between words located in different phrases.

1. LAST: Given a dependent phrase and its head phrase in the original annotation, set the head of the last word in the dependent phrase to the last content word in the head phrase. Note, we only apply this rule if the head phrase consists of a content word followed by zero or more function words, followed by an optional punctuation symbol.

[3] In addition to phrase dependency annotations, the NTC also contains predicate-argument and coreference tags that are useful for deriving reliable word dependencies.



word-based dependency corpus (partially annotated)

ID	head	word	rule
01	02	党内/noun	CF
02	03	の/part.	LAST
03	04	議論/noun	CF
04	19	は/part.	LAST
05	15	「/symbol	PAREN
06		保守/noun	
07		二/noun	
08		党/suff.	
09	10	論/noun	CF
10		は/part.	
11	12	よろし/adj.	INFLECT
12		く/infl.	
13	14	な/adj.	INFLECT
14	15	い/infl.	FUNCT
15		」/symbol	
16	17	と/part.	LAST
17	18	い/verb	INFLECT
18	19	う/infl.	LAST
19		もの/noun	
20	21	だ/aux.	FUNCT
21	–	。/symbol	

Figure 3: An example of word-based dependencies derived from phrase-based dependencies for a sentence.

2. PAREN: Set the head of a left parenthesis (or left bracket) word to the first right parenthesis (or right bracket) that follows it in the sentence.

The last four rules are intra-phrase rules that are concerned with the dependencies between words in the same phrase. The following rules were found to be effective.

3. FFS: If a phrase consists of zero or more content words, function words, or punctuation symbols followed by a sequence of two function words and a punctuation symbol, then set the head of the first function word in the sequence to the second function word and the head of the second function word to the punctuation symbol.

4. CF: If a phrase consists of zero or more content words followed by a sequence of a content word and a function word, then set the head of the content word to the function word.

5. INFLECT: If a word that is inflected in Japanese (verb, auxiliary verb, or adjective[4]) is followed by an inflection, the first word depends on the inflection.

6. FUNCT: If a function word is followed by a punctuation symbol, set the head of the function word to the punctuation symbol.

4 Evaluation

As an evaluation of our parser, we measured parsing accuracies of several systems on test corpora in two domains: one is a general domain in which a corpus fully annotated with word boundary and dependency information is available, and the other is a target domain assuming an adaptation situation in which only a partially annotated corpus is available for quick and low-cost domain adaptation.

4.1 Experimental Settings

In the experiments we used example sentences from a dictionary (Keene et al., 1992) as the general domain data, and newspaper articles as the target domain data. We used business newspaper articles (Nikkei), similar to the Wall Street Journal, for the target domain test set. For the domain adaptation experiments, we used the partially-annotated corpus mentioned in Section 3.3 as a

[4] In Japanese there are two types of adjectives, *i*-type adjectives and *na*-type adjectives. Both types are inflected.

Table 1: Sizes of Corpora.

ID	source	usage	#sentences	#words	#chars
EHJ-train	example sentences	training	11,700	145,925	197,941
EHJ-test	from a dictionary	test	1,300	16,348	22,207
NTC-train	newspaper articles	training	34,712	1,045,328	1,510,618
NKN-test	newspaper articles	test	1,002	29,038	43,695

NTC-train is a partially annotated corpus derived from phrase-based dependency annotations.

target domain training corpus. This corpus consists of newspaper articles that are similar to the target domain test set.

Usages and specifications of the various corpora are shown in Table 1. All the sentences are segmented into words manually and all the words are annotated with their heads manually. The Japanese data provided by the CoNLL organizers (Buchholz and Marsi, 2006) are the result of an automatic conversion from phrase (*bunsetsu*) dependencies. For a more appropriate evaluation we have prepared a word-based dependency data set.

The dependencies have no labels because almost all nouns are connected to a verb with a case marker and many important labels are obvious. The words are not annotated with POS tags, so we used a Japanese POS tagger, KyTea (Neubig et al., 2011), trained on about 40k sentences from the BCCWJ (Maekawa, 2008).

For the general domain experiments we compared the following systems, using projective parsing algorithms for training because of the assumptions about Japanese parsing outlined in Section 2.1.

1. **Malt**: Nivre et al. (2006)'s MaltParser, using Nivre's arc-eager algorithm and the option for strict root handling.

2. **MST**: McDonald et al. (2005)'s MST Parser, using k-best parse size with k=5.

3. **PW**: Our system, where pointwise estimation is used to estimate dependencies. Stochastic gradient descent training was used to train log-linear models.

4.2 With a Fully Annotated Training Corpus

For the first experiment, we measured the accuracy of each system on an in-domain test set when training on a fully annotated corpus. The results are shown in Table 2. Malt and MST have similar accuracy. PW has slightly higher accuracy than both of these systems, but the difference in accuracy is not statistically significant. We also measured the training time and the parsing speed of each system. Table 3 shows the results. From this table, first we see that MST is much slower than Malt, as is well known. Our method, however, is much faster than MST and the parsing speed is approximately the same as the shift-reduce-based Malt.

Theoretically the training time of our method is proportional to the number of annotated dependencies. In line with Kudo and Matsumoto (2002), we make two assumptions about Japanese dependency parsing. First, because Japanese is a head-final language we assume that every word except the final one in a sentence depends on one of the words located to its right. Second, we assume that all dependencies are projective, in other words that edges in the dependency tree do not cross each other. These assumptions limit the number of candidate heads for a word, reducing the training time.

Because all parsers were trained with projective algorithms, the first assumption is most likely the main reason for the difference in training times between PW and MST. For other languages where possible heads can be located both to the left and right of a word, we expect training times to increase. Our pointwise approach can be extended to handle these languages by changing the constraint on heads from $j > i$ to $j \neq i$ for all $d_i = j$. This is an important direction for future work now that we have confirmed that this approach is effective for Japanese.

We performed a second experiment in the general domain to measure the impact of the training corpus size on parsing accuracy. To make smaller training corpora, we set a fixed number of dependency annotations and then sequentially selected sentences from EHJ-train until the desired number of dependency annotations were collected. The re-

Table 2: Parsing Accuracy on EHJ-test.

method	EHJ-test
Malt	96.63%
MST	96.67%
PW	96.83%

All systems were trained on EHJ-train.

Table 3: Training Time and Parsing Speed.

method	training time	parsing speed
Malt	14[s]	1.3[ms/sent.]
MST	1901[s]	32.7[ms/sent.]
PW	125[s]	2.8[ms/sent.]

All systems were trained on EHJ-train and tested on EHJ-test. The machine used had a 3.33GHz processor and 12GB of RAM.

sults are shown in Figure 4. Though PW achieves the highest accuracy when the full training corpus is used, Malt has higher accuracy than both of the MST-based systems when the training corpus is one-third or less the size of the full training corpus. It can also be shown that both MST-based systems improve at a similar rate for all sizes of training corpora.

4.3 Domain Adaptation with a Partially Annotated Training Corpus

Tasks that make use of parsers, such as machine translation (Yamada and Knight, 2001), often require word-based models. However, because phrase-based approaches have traditionally been used for Japanese dependency parsing (Kudo and Matsumoto, 2002; Sassano and Kurohashi, 2010), word-based linguistic resources for Japanese are scarce. Preparing the fully annotated corpora required by existing word-based parsers such as McDonald et al. (2005)'s MST Parser is an expensive and laborious task.

Our parser attempts to address these problems by introducing a word-based framework for dependency parsing that can use partially annotated training data. Partial annotation is one way to efficiently make use of existing resources in the target domain without incurring high annotation costs.

We used each rule described in Section 3.3 individually to convert the annotations in the NTC-train and produce a pool of word-based dependencies. We then selected 5k of those dependencies to add to EHJ-train, and measured the results on NKN-test. We also used all rules simultaneously to produce word-based dependencies and measured the results in the same way as the individual rules. The total size of the partial annotation pool produced by using all rules was 248,148 dependencies out of 1,010,648 annotation candidates (not counting the last word of sentences, which has no dependency). The baseline case only used the EHJ-train with no partial annotations from the pool. The results are shown in Figure 5.

It can be seen that the LAST rule is the most effective, followed by the PAREN rule. This suggests that the long-distance dependencies provided by these rules are more useful for domain adaptation than the short-distance dependency information that the intra-phrase rules provide.

Combining all of the rules increases the accuracy on NKN-test to 88.44%, an increase of 2.75% over the baseline. This combination of rules results in lower accuracy gains than the sum of the gains from individual rules because different rules may convert the same phrase dependencies. These results show that our pointwise approach allows for effective use of existing target domain resources and increased parsing accuracy in the target domain through partial annotation.

5 Related Work

There has been a significant amount of work on how to utilize in-domain data to improve the accuracy of parsing. The majority of this work has focused on using unlabeled data in combination with self-training (Roark and Bacchiani, 2003; McClosky et al., 2006) or other semi-supervised learning methods (Blitzer et al., 2006; Nivre et al., 2007; Suzuki et al., 2009).

Roark and Bacchiani (2003) also present work on supervised domain adaptation, although this focuses on the utilization of an already-existing in-domain corpus.

There has also been some work on efficient annotation of data for parsing (Tang et al., 2002; Osborne and Baldridge, 2004; Sassano and Kurohashi, 2010). Most previous work focuses on picking efficient sentences to annotate for parsing, but Sassano and Kurohashi (2010) also present a method for using partially annotated data with deterministic dependency parsers, which can be trivially estimated from partially annotated data.

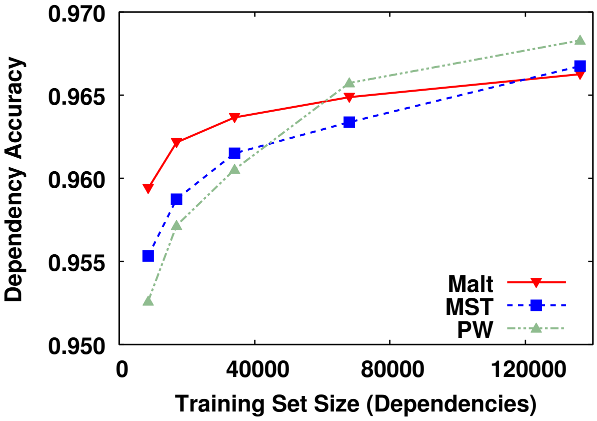


Figure 4: Comparison of parsing accuracy for different parsers.

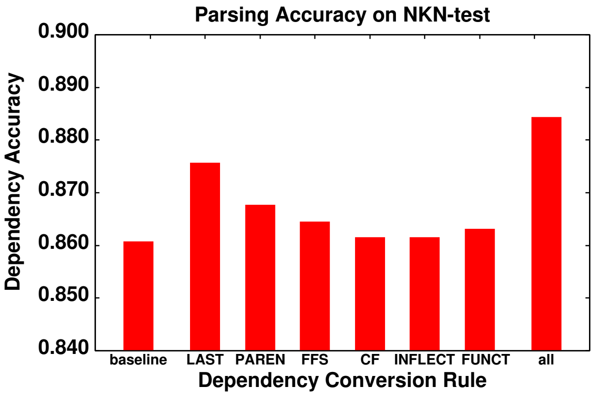


Figure 5: Parsing Accuracy on NKN-test.

Other recent work (Spreyer and Kuhn, 2009; Spreyer et al., 2010) has shown how both Nivre et al. (2006)'s MaltParser and McDonald et al. (2005)'s MST can be adapted to use partially annotated training data.

Traditional MST parsers such as McDonald et al. (2005)'s use structured prediction methods. Wang et al. (2007) showed that local classification methods can be used to train structured predictors. Their approach also uses "dynamic" features, where the predictions for some surrounding edges are used as features when estimating a possible edge between a dependent and head word.

Our parser also makes use of local classification methods for training, but in contrast to Wang et al. (2007) we take a pointwise approach based on the assumption that edge scores can be estimated independently. This work follows in the thread of Liang et al. (2008) and Neubig et al. (2011), who demonstrated that these assumptions can be made without a significant degradation in accuracy for POS tagging. Here we demonstrated that the same approach can be used for MST-based dependency parsing.

6 Conclusion

We introduced an MST parser that evaluates the score for each edge in a dependency tree independently, which allows for the use of partially annotated corpora in training. We demonstrated that target domain data annotated in this way can be combined with available general domain data to increase parsing accuracy in the target domain.

We evaluated state-of-the-art dependency parsers on a Japanese dependency parsing task, and found that our parser achieves accuracy comparable to that of a traditional MST parser. Additionally, the training and parsing speed of our parser is much faster than the traditional one, which allows it to be used for active learning in a real-world domain adaptation situation.

References

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

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 120–128.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164.

Daniel Gildea. 2001. Corpus variation and parser performance. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 167–202.

Ryu Iida, Mamoru Komachi, Kentaro Inui, and Yuji Matsumoto. 2007. Annotating a Japanese text corpus with predicate-argument and coreference relations. In *Proceedings of the Linguistic Annotation Workshop*, pages 132–139.

Donald Keene, Hiroyoshi Hatori, Haruko Yamada, and Shouko Irabu. 1992. *Japanese-English Sentence Equivalents (in Japanese)*. Asahi Press, Electronic book edition.

Taku Kudo and Yuji Matsumoto. 2002. Japanese dependency analysis using cascaded chunking. In *Proceedings of the Sixth Conference on Computational Natural Language Learning*, volume 25, pages 1–7.

Percy Liang, Hal Daumé III, and Dan Klein. 2008. Structure compilation: trading structure for features. In *Proceedings of the 25th International Conference on Machine Learning*, pages 592–599.

Kikuo Maekawa. 2008. Balanced corpus of contemporary written Japanese. In *Proceedings of the 6th Workshop on Asian Language Resources*, pages 101–102.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Reranking and self-training for parser adaptation. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics*, pages 337–344.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the 2005 Conference on Empirical Methods in Natural Language Processing*, pages 523–530.

Yusuke Miyao, Kenji Sagae, Rune Saetre, Takuya Matsuzaki, and Jun'ichi Tsujii. 2009. Evaluating contributions of natural language parsers to protein-protein interaction extraction. *Bioinformatics*, 25(3):394–400.

Toshiaki Nakazawa and Sadao Kurohashi. 2008. Linguistically-motivated tree-based probabilistic phrase alignment. In *In Proceedings of the Eighth Conference of the Association for Machine Translation in the Americas (AMTA2008)*.

Graham Neubig and Shinsuke Mori. 2010. Word-based partial annotation for efficient corpus construction. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation*.

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise prediction for robust, adaptable Japanese morphological analysis. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies Short Paper Track*.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of English text. In *Proceedings of the 20th International Conference on Computational Linguistics*.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*.

Joachim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*.

Miles Osborne and Jason Baldridge. 2004. Ensemble-based active learning for parse selection. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 89–96.

Slav Petrov, Pi-Chuan Chang, Michael Ringgaard, and Hiyan Alshawi. 2010. Uptraining for accurate deterministic question parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 705–713.

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

Manabu Sassano and Sadao Kurohashi. 2010. Using smaller constituents rather than sentences in active learning for Japanese dependency parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 356–365.

Kathrin Spreyer and Jonas Kuhn. 2009. Data-driven dependency parsing of new languages using incomplete and noisy training data. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 12–20.

Kathrin Spreyer, Lilja Øvrelid, and Jonas Kuhn. 2010. Training parsers on partial trees: a cross-language comparison. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation*.

Jun Suzuki, Hideki Isozaki, Xavier Carreras, and Michael Collins. 2009. An empirical study of semi-supervised structured conditional models for dependency parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 551–560.

Min Tang, Xiaoqiang Luo, and Salim Roukos. 2002. Active learning for statistical natural language parsing. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 120–127.

Yuta Tsuboi, Hisashi Kashima, Shinsuke Mori, Hiroki Oda, and Yuji Matsumoto. 2008. Training conditional random fields using incomplete annotations. In *Proceedings of the 22th International Conference on Computational Linguistics*.

Qin Iris Wang, Dekang Lin, and Dale Schuurmans. 2007. Simple training of dependency parsers via structured boosting. In *Proceedings of the TwentiethInternational Joint Conference on Artificial Intelligence*, pages 1756–1762.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical machine translation model. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 523–530.