A Pointwise Approach to Training Dependency Parsers from Partially Annotated Corpora

Daniel Flannery[†], Yusuke Miyao[††], Graham Neubig[†,††††] and Shinsuke Mori[†††]

We introduce a word-based dependency parser for Japanese 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.

Key Words: Dependency parsing, partial annotation, domain adaptation

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, Sagae, Saetre, Matsuzaki, and Tsujii 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, Chang, Ringgaard, and Alshawi 2010), and that in-domain data can be annotated to make up for this weakness.

In this paper, we propose a dependency parser for Japanese 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. We use the assumption that Japanese is a head-final language to simplify decoding by constraining the size of the search space. 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

[†]Graduate School of Informatics, Kyoto University
[††]National Institute of Informatics
[†††]Academic Center for Computing and Media Studies, Kyoto University
[††††]Now affiliated with the Nara Institute of Science and Technology

tasks (Tsuboi, Kashima, Mori, Oda, and Matsumoto 2008; Sassano and Kurohashi 2010; Neubig and Mori 2010), traditional parsers such as that of (McDonald, Pereira, Ribarov, and Hajič 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, Daumé III, and Klein 2008; Neubig, Nakata, and Mori 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, Øvrelid, and Kuhn 2010) has shown that training constraints can be relaxed to allow 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 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 McDonald et al. (2005)'s parser 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 McDonald et al. (2005)'s parser, while training and testing speeds are similar to those 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 a sequence of words $\boldsymbol{w} = \langle w_1, w_2, \ldots, w_n \rangle$ as input, 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 Dependency Parser

The parsing model we pursue in this paper is McDonald et al. (2005)'s edge-factored model. A score, $\sigma(\langle i, d_i \rangle, \boldsymbol{w})$, 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

$$\hat{\boldsymbol{d}} = \mathop{\mathbf{argmax}}_{\boldsymbol{d} \in D} \sum_{i=1}^{n} \sigma(\langle i, d_i \rangle, \boldsymbol{w}), \tag{1}$$

where D is the set of all possible spanning trees for the input sentence.

It is known that, given $\sigma(\langle i, d_i \rangle, \boldsymbol{w})$ 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 $\sigma(\langle i, d_i \rangle, \boldsymbol{w})$. McDonald et al. (2005) applied a perceptron-like algorithm that optimizes the score of entire dependency trees. However, we stick to pointwise prediction: $\sigma(\langle i, d_i \rangle, \boldsymbol{w})$ is estimated for each w_i independently. Any variety of machine-learning-based classifiers can be applied to the estimation of $\sigma(\langle i, d_i \rangle, \boldsymbol{w})$, because it is essentially an n-class classification problem.

We define the edge score as a probability, $\sigma(\langle i, d_i \rangle, \boldsymbol{w}) = \log p(d_i)$, and estimate a log-linear model (Berger, Della Pietra, and Della Pietra 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 $\boldsymbol{\theta} = \langle \theta_1, \theta_2, \ldots, \theta_m \rangle$.

[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		NULL, NULL, 政府						つなが, る, と	
F5		投資, に, つなが						し, NULL, NULL	
F6		NULL, NULL, noun						verb, infl., part.	
F7		noun, part., verb						verb, NULL, NULL	

The second word, the 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.

Fig. 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).

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, both individually and as combination features, for $\boldsymbol{\phi}$.

F1: The distance $j - i$ between a dependent word w_i and its candidate head w_j.

F2: The surface forms w_i and w_j.

F3: The parts-of-speech of w_i and w_j.

F4: The surface forms of up to three words to the left of w_i and w_j.

F5: The surface forms of up to three words to the right of w_i and w_j.

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, the case marker は (*subj.*), has been annotated with its head, the verb 歓迎 (welcomes).

Using 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, which is important as recent work on active learning for word segmentation and POS tagging (Neubig et al. 2011) has shown the importance of learning speed for active learning strategies.

2.3 First-Order and Second-Order Features

McDonald et al. (2005)'s original approach is called a first-order formulation because features are only defined over the dependent and head words forming a single edge. The main features used are the surface forms and POS tags of the dependent and head, and distance between them. They also incorporate local context information by defining features on words to the immediate left and right of both the dependent and head, for a window size of three words. Similarly, they make use of broader context information by defining features on the POS tags of words that occur between the dependent and head.

McDonald and Pereira (2006) later extended the first-order approach of McDonald et al. (2005) to a second-order approach, where information about adjacent edges is also used as features. In this new formulation, the score of the tree is factored into the sum of adjacent edge pair scores instead of the sum of individual edge scores. Because up to two adjacent dependency edges from the same head are considered when computing an edge pair score, this has the effect of conditioning on the last dependent chosen for the head.

In contrast to the second-order formulation described above, the proposed method sticks to first-order features but refers to a larger window of surrounding words for both the dependent and head. Up to three words in each direction are considered, resulting in a window size of seven words. This allows us to pick up context information regardless of whether an adjacent edge exists for a head. Sassano and Kurohashi (2009) showed that this kind of context information can also be useful for phrase-based Japanese dependency parsing.

There are three main motivations for our pointwise approach. First, we wish to avoid feature sparsity in the training data by restricting ourselves to first-order features. Second, we want to enable our parser to be trained from partially annotated corpora, where only some dependencies in a sentence are annotated. Finally, we seek to reduce the amount of time necessary for training. We will show in Section 4.2 that for a Japanese dependency parsing task, the proposed method achieves reasonably good parsing accuracy.

2.4 Solution Search

The target of our experiments is written Japanese, which is a head-final language. In line with Uchimoto, Sekine, and Isahara (1999), we assume that in Japanese dependencies go from left to right and that every word except for the last one in a sentence depends on exactly one other word. Thus we assume that $d_i > i$ 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.

In contrast to Uchimoto et al. (1999) we do not make the assumption that dependencies do not cross, because even in written Japanese such dependencies may occur in informal contexts. Our implementation does not enforce this projectivity constraint on dependencies, so it can handle non-projective dependencies in the training data which satisfy the head-final assumption.

This head-final assumption does not hold for spoken Japanese and languages such as English, but it is easy to extend our implementation to handle these cases. Specifically, this can be done by changing the constraint on heads from $d_i > i$ to $d_i \neq i$ and using Chu-Liu/Edmonds' algorithm to ensure that no loops of dependencies are created while building the maximum spanning tree for the sentence. This algorithm has the additional benefit of handling all types of non-projective dependencies. Because the proposed method for learning feature weights from partially annotated data does not depend on the parsing algorithm, different parsing algorithms could also be used, for example to enforce projectivity constraints.

2.5 Japanese Dependency Parsing

Kudo and Matsumoto (2000) also proposed a probabalistic parser for Japanese which uses the assumption that edges can be estimated independently. Their approach uses *bunsetsu* (chunks) instead of words and is limited to projective dependencies, but is otherwise similar to the proposed method. The key difference is how context information is used in their feature set. During both training and parsing, information about dependencies for chunks between a candidate dependent and head is used for determining whether a dependency exists between them. These are called "dynamic features" because they are updated dynamically during parsing as dependencies in the sentence are estimated, in contrast to "static" features like POS tags, which depend only on the input string and do not change. While features based on surrounding dependencies are trivial to use during training, such features are difficult to use during parsing because the structure for the dependency tree is not known. Dynamic features are updated as the dependency tree structure for the sentence is built incrementally, allowing dependencies that have been finalized to be used as information for those that have not. Kudo and Matsumoto (2000) conclude that models with these dynamic features consistently outperform those without them.

In contrast, our approach does not use estimated values such as these dynamic features when determining whether a dependency exists between a given pair of words when parsing. Instead, our feature set uses a larger number of static features to capture context information. When evaluating a potential dependency between a dependent and a head word, surface forms and POS tags in a window of seven words for both are used as features. Dependencies are estimated independently to enable training from partial annotation.

Because Kudo and Matsumoto (2000)'s model assumes that edges are independent of each other, it is theoretically possible to adapt it so that it can use partially annotated training data. Dynamic features based on chunks between a candidate dependent and head chunk are an important part of their approach, so to annotate a single dependency for a pair of chunks an annotator would have to annotate the heads for chunks in between them. This makes partial annotation time-consuming for chunks which are not adjacent. Sassano (2005) showed how partial annotation can be used for Japanese dependency parsing, but only considered partial annotations consisting of adjacent chunks for this reason.

In our model it is sufficient to annotate individual dependencies between words, so even long-distance dependencies are easy to use as partial annotations. We leave the problem of finding an informative criterion for selecting annotations for Japanese dependency parsing as future work.

3 Domain Adaptation for Dependency 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, 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, $\sigma(\langle i, d_i \rangle, \boldsymbol{w})$, are estimated for each w_i independently. This means that the estimation of $\sigma(\langle i, d_i \rangle, \boldsymbol{w})$ 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 dependency that w_i depends on w_j, a discriminative classifier can be trained by regarding $d_i = j$ as a positive sample and $d_i = j'$ where $j' \neq j$ as negative samples.

In the case of Japanese parsing, because $j > i$ for all $d_i = j$, negative samples are $d_i = j'$ where $j' \neq j$ and $j' > i$. 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, Komachi, Inui, and Matsumoto 2007) to create a small

[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

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

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.

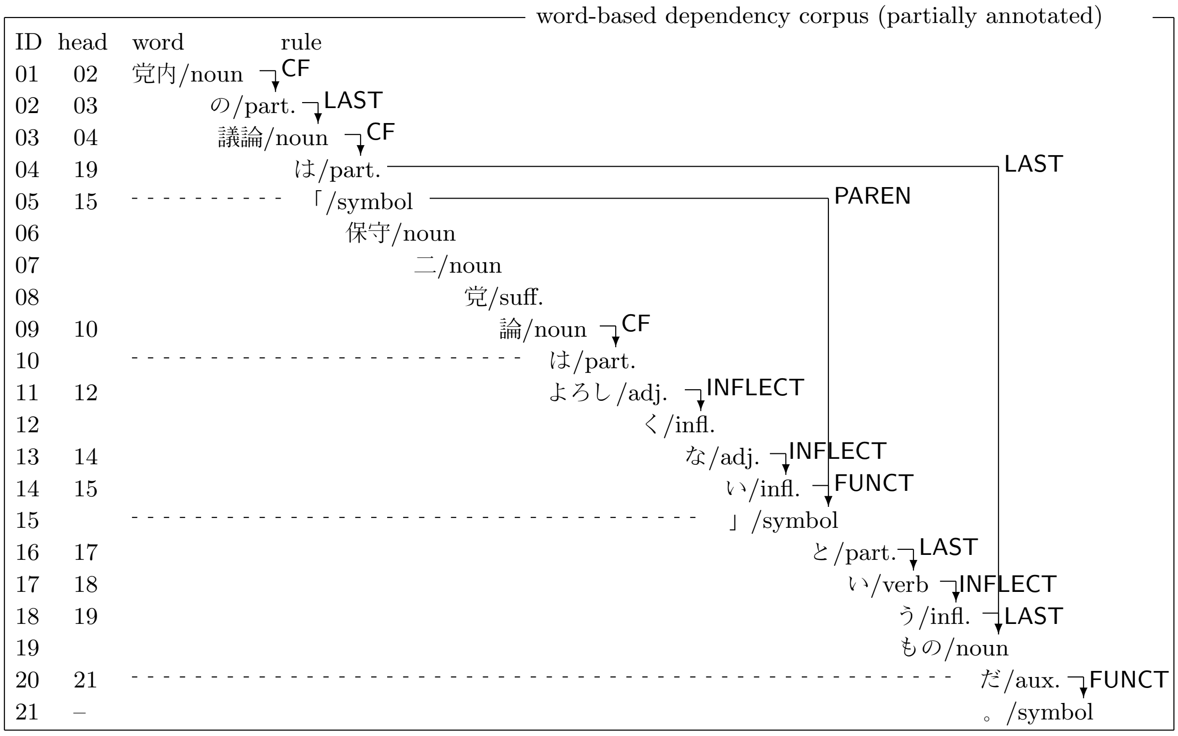
The first two rules are inter-phrase rules, which are concerned with the relationship between words located in different phrases.

(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.

(2) PAREN: Set the head of a left parenthesis (or left bracket) 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]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	

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

(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]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.

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, Hatori, Yamada, and Irabu 1992) as the general (source) domain data, and business newspaper articles (Nikkei), similar to the Wall Street Journal, for the target domain test set. Compared to the dictionary examples, the newspaper articles use a more formal writing style, specialized vocabulary, and longer sentences. Thus, the domains of the two corpora are different enough to justify domain adaptation techniques.

For the domain adaptation experiments, we used the partially-annotated corpus mentioned in Section 3.3 as a 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, except for NTC-train. 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

Table 2 Parsing Accuracy on EHJ-test.

method	EHJ-test
Malt	96.85%
2nd-order MST	96.74%
1st-order MST	96.65%
EDA	96.83%

All systems were trained on EHJ-train. Note that each system uses a different feature set.

from the Balanced Corpus of Contemporary Written Japanese (BCCWJ) (Maekawa 2008).

For the general domain experiments we compared the following systems.

(1) **Malt**: Nivre, Hall, and Nilsson (2006a)'s MaltParser. We chose the projective arc-eager algorithm and HEAD option (`-pp head`) to projectivize the training data because these settings[5] achieved the best performance for Japanese on the CoNLL-X shared task (Nivre, Hall, Nilsson, Eryiğit, and Marinov 2006b).

(2) **1st-order MST**: McDonald et al. (2005)'s MST Parser, using the options for first-order parsing, non-projective decoding, and k-best parse size with k=1. We chose the non-projective decoding option for a fairer comparison with the proposed method, which does not enforce projectivity constraints.

(3) **2nd-order MST**: The same as 1st-order MST, but with the option for second-order parsing (McDonald and Pereira 2006).

(4) **EDA**: Our system ("Easily adaptable Dependency Analyzer"[6]), which uses pointwise estimation and first-order features to estimate dependencies. We used stochastic gradient descent for training.

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. Our goal is to see how our method performs in comparison to state-of-the art multilingual parsers when parsing Japanese. The results are shown in Table 2. All systems achieve high accuracy on this task, and no differences between

[5]See `http://www.maltparser.org/conll/conllx/` for details on the optimal hyperparameter settings for Japanese. We also chose LIBLINEAR (with MaltParser's default choice of a multi-class SVM) as the learner instead of LIBSVM, as recommended in the MaltParser optimization guide available at `http://www.maltparser.org/guides/opt/quick-opt.pdf` (links accessed April 2012).

[6]Available at `http://plata.ar.media.kyoto-u.ac.jp/tool/EDA/`

Table 3 Training Time and Parsing Speed.

method	training time	parsing speed
Malt	23[s]	1.4[ms/sent.]
2nd-order MST	3911[s]	35.9[ms/sent.]
1st-order MST	1236[s]	32.7[ms/sent.]
EDA	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.

systems were statistically significant ($p > 0.05$, according to Pearson's χ^2 test). Malt and EDA have similar accuracy, while both variations of MST have only slightly lower accuracy.

Our model factors the score for a dependency tree into the sum of individual edge scores in the same way as the 1st-order MST model, so we expected their performance to be close. The richer feature set of our method is most likely the reason for the small difference in performance between EDA and 1st-order MST. This result shows that our pointwise approach achieves comparable accuracy on Japanese to that of state-of-the art parsers while allowing for much more flexible use of language resources. This flexibility is very important in practical situations.

In contrast, Malt and 2nd-order MST both use a history-based feature set, which incorporates more context than the edge-factored approaches of EDA and 1st-order MST. In the case of Malt, the partially built dependency structure of a sentence is used as features (Nivre et al. 2006a), which are similar to the "dynamic features" used by Kudo and Matsumoto (2000) and discussed in Section 2.5. As discussed in Section 2.3, 2nd-order MST factors the sentence into a set of edge pairs instead of individual edges.

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 both 1st-order and 2nd-order versions of MST are much slower than Malt, as is well known. 2nd-order MST takes more than three times as long to train as 1st-order MST, but their parsing speed is almost identical. The training time of our method is in between Malt and both versions of MST – while it is much slower than the shift-reduce based Malt, this result shows that our method is fast enough to be used for active learning. Training speed is crucial for active learning because the annotator must wait while the model is retrained after each round of annotation. Neubig et al. (2011) demonstrated the effectiveness of the pointwise approach in a realistic active learning scenario.

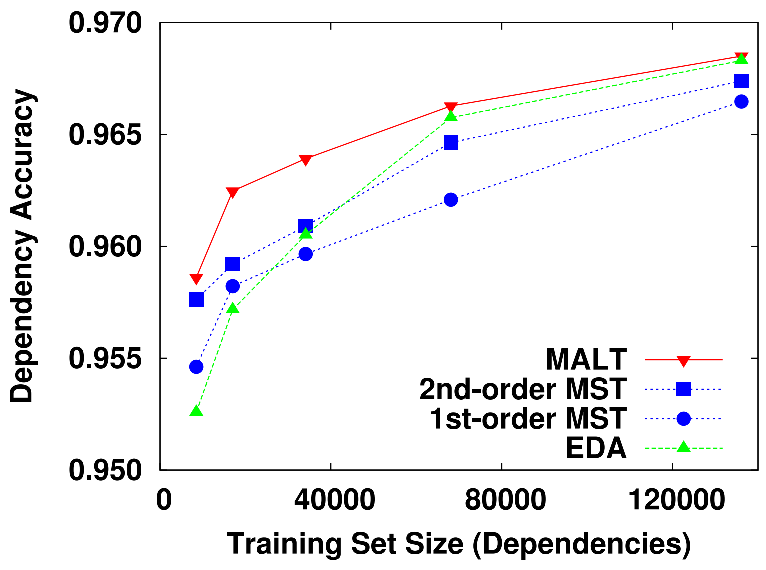


Fig. 4 Comparison of parsing accuracy for different parsers.

Theoretically the training time of our method is proportional to the number of annotated dependencies. The assumptions outlined in Section 2.4 are most likely the main reason for the difference in training times between EDA and the two versions of MST. For other languages where possible heads can be located both to the left and right of a word, we expect training and parsing 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 results are shown in Figure 4. For smaller training corpora Malt outperforms all other systems, but the difference is less pronounced when at least half of the training corpus is used. The proposed method's performance lags behind the other systems when little training data is available, but is comparable when at least half of the training data is used. While 2nd-order MST outperforms 1st-order MST, the difference is not pronounced. This is probably because dependency arcs in this data set always point to the right – in a standard dependency parsing task where arcs may go in either direction, we expect 2nd-order MST to consistently outperform 1st-order MST.

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 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 Comparison with a Phrase-based Dependency Parser

Because the phrase-based approach is the most commonly used in work on Japanese dependency parsing, we also compared the performance of our word-based method to a traditional phrase-based method, the cascaded chunking approach of Kudo and Matsumoto (2002). A direct

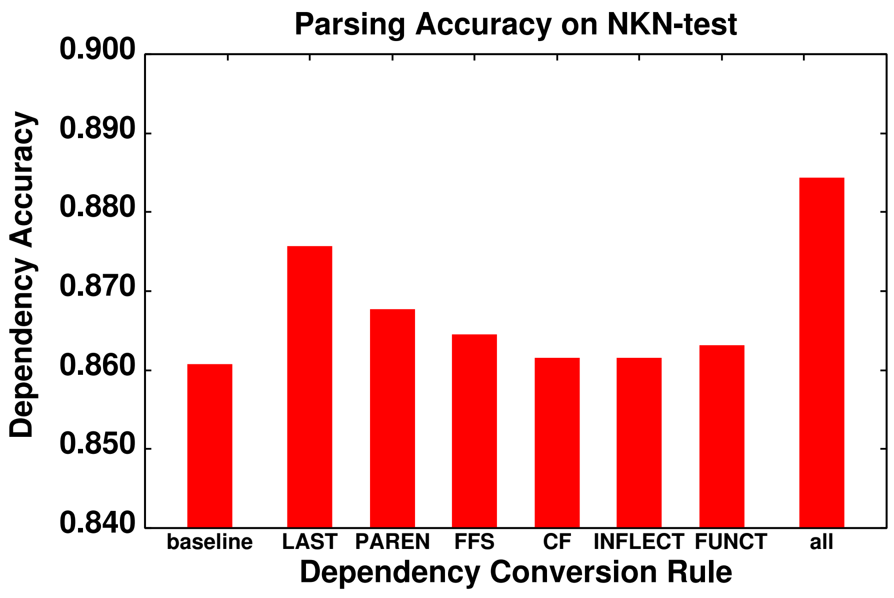
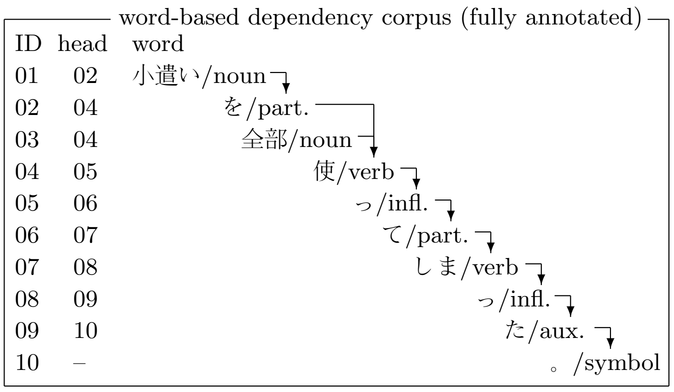


Fig. 5 Parsing Accuracy on NKN-test.

comparison is difficult because it would require data annotated with both word and phrase dependencies. However, if a corpus is annotated with word dependencies and POS tags, and we have an assignment of words to phrases, it is possible to use heuristics to estimate the corresponding phrase dependency annotations. This is because the information provided by word dependencies is more fine-grained than the information provided by phrase dependencies, and the former can be seen as a superset of the latter. Phrase dependencies can be viewed as modeling only the relationships among each phrase's key words, ignoring any dependency information between words in the same phrase. Figure 6 shows a sentence annotated with word-based dependencies and POS tags which will be used to create phrase dependency annotations.

5.1 Converting Word Dependencies to Phrase Dependencies

The conversion is a two-step process: we first use only POS tags to group words into phrases, and then we use dependencies between words in different phrases to assign phrase dependencies. The second step is straightforward because of our assumption that Japanese is head-final. Just as in the case of word dependencies, when estimating phrase dependencies we only consider heads which occur to the right of their dependents and do not allow non-projective dependencies. Therefore a single scan through all phrases in sentence order is sufficient to assign their heads. For a given phrase, we simply scan through each of its words in order, checking to see if the



word-based dependency corpus (fully annotated)

ID	head	word
01	02	小遣い/noun
02	04	を/part.
03	04	全部/noun
04	05	使/verb
05	06	っ/infl.
06	07	て/part.
07	08	しま/verb
08	09	っ/infl.
09	10	た/aux.
10	–	。/symbol

Fig. 6 An example of word-based dependency annotation for a sentence.

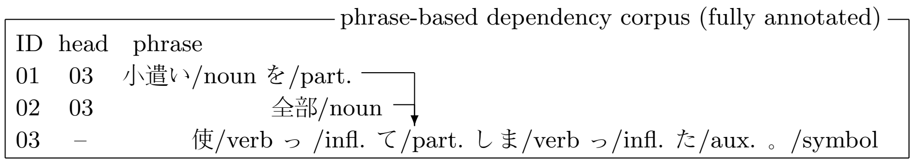
word's head belongs to a different phrase than the current one. If it does, we set the current phrase's head to the one containing the head word and then begin processing the next phrase in the sentence[7]. We stop processing when we reach the last phrase, because by convention its head is the root.

The first step is more difficult and required us to develop some heuristic rules. To formulate these rules, we made use of the idea of function words and content words described in Section 3.3. We used the same set of function words, but we added pronouns and adverbs to the set of content words. This is because we took a bottom-up approach to building phrases where we incrementally add words to a phrase, deciding whether or not to insert a phrase boundary after each word.

The basic procedure is as follows. We begin with an empty phrase and then examine each word in order, considering whether or not to add it to the current phrase. We first check to see whether the POS tag for the word belongs to the set of function words. If the word's tag is not in the set of function words, we add the word to the current phrase. If the word is a prenominal adjective, adverb, conjunction, or adverbial noun[8], we insert a phrase boundary after the word. We refer to this set of words as *boundary words* because they are likely to indicate a phrase boundary. On the other hand, if the word is in the set of function words we add it to the current phrase and examine the next word, adding it to the current phrase only if it is also a function word. However, if the next word is a boundary word or there are no more function words remaining in the sentence we insert a phrase boundary between the first word and the next word.

[7]We always choose the first available head for phrase to avoid possible conflicts, though in practice these rarely occurred.

[8]Adverbial nouns are nouns which can also function as adverbs by indicating a frequency or amount.



phrase-based dependency corpus (fully annotated)

```
ID  head  phrase
01  03    小遣い/noun を/part.
02  03                  全部/noun
03  –             使/verb っ/infl. て/part. しま/verb っ/infl. た/aux. 。/symbol
```

Fig. 7 An example of phrase-based dependencies derived from word dependencies.

The procedure outlined above will give us a coarse-grained phrase segmentation for the sentence, but there are some edge cases which require further segmentation of these phrases. We identified four of these cases and checked each phrase created by the original phrase segmentation against them. Note that in both the initial coarse-grained phrase segmentation and the subsequent fine-grained segmentation, before inserting a phrase boundary we check the surrounding words to avoid splitting constructions commonly treated as a single phrase in Japanese.

(1) When a phrase contains a right quote or parenthesis followed by the corresponding left quote or parenthesis, insert a phrase boundary between them.

(2) Insert a phrase boundary between a function word that is immediately followed by a content word.

(3) When there is a noun or pronoun immediately followed by a verb, insert a phrase boundary between them. We do not apply this rule when the verb is する (to do) or a related verb, because this indicates that the noun is acting as a verbal noun.

(4) When dictionary form of the verb する (to do) is followed by a noun, insert a phrase boundary between the two words.

Figure 7 shows the result of applying the steps outlined above to convert the word dependencies shown in Figure 6 to phrase dependencies. After the coarse-grained phrase segmentation the words in phrases 02 and 03 are initially grouped into a single phrase. This phrase corresponds to case (3) above because it contains the noun 全部 (all) followed by a verb, so a phrase boundary is inserted between these two words.

5.2 Evaluation on a Phrase-based Test Set

After both corpora were annotated with phrase dependencies, we performed an experiment to measure the unlabeled phrase dependency accuracy of the cascaded chunking method (CC) and the proposed method (EDA). First, we converted the gold word-based dependencies for EHJ-test to phrase-based dependencies. For EDA, we trained a parser as described in Section 4.2 and then converted the parser's output on the word-based version of EHJ-test to phrase-based dependencies. The same procedure was used to convert both the gold dependencies and the parser output for the test set, ensuring that the phrase segmentation was consistent between them. It should be noted that the training and test data sets consist of example sentences from a dictionary, which in general are much shorter than sentences from common domains such as newspaper articles. Thus, the results on these types of data sets are likely to differ from the ones we report below.

For CC, we first converted EHJ-train to phrase-based dependencies and then used Kudo and Matsumoto (2002)'s implementation, CaboCha[9], to train a dependency parsing model. We chose 3 as the degree of the polynomial kernel since this setting has been demonstrated to be effective for Japanese dependency parsing (Kudo and Matsumoto 2000, 2002). We then used that model to parse the phrase-based version of EHJ-test created from the gold word dependencies. To ensure that the phrase segmentation and POS tags were consistent for the test set, we did not use the phrase segmentation or POS tagging features of their implementation.

Kudo and Matsumoto (2002)'s cascaded chunking approach uses the base feature set of Kudo and Matsumoto (2000)'s probabalistic model (discussed in Section 2.5) and adds additional dynamic features based on the chunks that modify the dependent chunk and the chunk which the head chunk modifies. They report that the cascaded chunking model requires fewer training examples and is thus much faster to train than the probabalistic model, which uses all candidate dependencies as training data. This is because the cascaded chunking model uses heuristics to prune exceptional dependencies (where possible correct heads are not selected because better one exists in the same sentence or because of projectivity constraints) from the training data. The training time for the proposed method is theoretically longer than that of the cascaded chunking method, because it uses a smaller unit (words instead of chunks) and uses all candidate dependencies as training data in the same way as the probabalistic model. However, for tasks like machine translation which require smaller units than chunks, the fine-grained dependency information of our approach is worth the additional training time.

The results are shown in Table 4. Though EDA's parsing speed is reasonably fast, CC's is much faster. It can also be seen that EDA outperforms CC by a small margin in terms of parsing accuracy[10]. Even though several of the dependencies between words may be obvious, the word-based dependency annotation provides us with richer information about the structure of the sentence than phrase dependencies. One possible cause for the difference in accuracy between CC and EDA is the POS tagging standard. CC is designed to use a detailed set of fine-grained POS tags, where broad categories such as nouns and verbs are further separated

[9]Available at `http://code.google.com/p/cabocha/` (accessed November 2011).
[10]This improvement in accuracy is statistically significant, with $p < 0.05$.

Table 4 *Bunsetsu* Parsing Accuracy on EHJ-test.

method	training time	test set size	accuracy	parsing speed
CC (Kudo and Matsumoto 2002)	222[s]	6,024[*bunsetsu*]	93.68%	0.14[ms/sent.]
EDA	145[s]	16,348[words]	94.41%	2.6[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.

into several subcategories. We use Maekawa (2008)'s POS tagging standard, which defines both coarse-grained and fine-grained tags. However, we only make use of the coarse-grained tags because these are more likely to be available in a realistic domain adaptation situation. Tagging words with fine-grained tags requires annotators to have experience with the tagging standard in addition to domain knowledge, which limits the number of potential annotators. Sticking to coarse-grained tags reduces the burden on annotators. Thus CC's accuracy suffers on this "poor" feature set because fine-grained POS tags are not available. Another possible cause for the differing accuracy may be the difference in granularity between word-based and phrase-based segmentation. For the same training data, there will be more examples of word dependencies than phrase dependencies.

6 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, Charniak, and Johnson 2006) or other semi-supervised learning methods (Blitzer, McDonald, and Pereira 2006; Nivre, Hall, Kübler, McDonald, Nilsson, Riedel, and Yuret 2007; Suzuki, Isozaki, Carreras, and Collins 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, Luo, and Roukos 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. Other recent work (Spreyer and Kuhn 2009; Spreyer et al. 2010) has shown how both Nivre et al. (2006a)'s MaltParser and McDonald et al. (2005)'s MST Parser can be adapted to use partially annotated training data.

Traditional parsers such as McDonald et al. (2005)'s use structured prediction methods. Wang, Lin, and Schuurmans (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 word segmentation and POS tagging. Here we demonstrated that the same approach can be used for dependency parsing.

7 Conclusion

We introduced a 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 source domain data to increase parsing accuracy in the target domain. We also showed how partial annotation can be leveraged to make use of corpora in different formats when a full conversion is not feasible.

In our evaluation on a Japanese dependency parsing task we found that our parser delivers accuracy comparable to that of state-of-the-art dependency parsers that use much more complex models, and has parsing and training speeds that are fast enough to allow for rapid domain adaptation. On a phrase-based Japanese dependency parsing task, our word-based parser slightly outperformed a traditional phrase-based parser. While our parser could not match the fast parsing speed of the traditional one, the training speeds and accuracy of both were comparable. The increased flexibility of simpler parsing models often comes at the price of decreased accuracy, but these results show that a simple model can be used to enable flexible domain adaptation without sacrificing accuracy.

Acknowledgment

This work was supported by Grant-in-Aid for Scientific Research of the government of Japan (23500177).

Reference

Berger, A. L., Della Pietra, V. J., and Della Pietra, S. A. (1996). "A maximum entropy approach to natural language processing." *Computational Linguistics*, **22** (1), pp. 39–71.

Blitzer, J., McDonald, R., and Pereira, F. (2006). "Domain Adaptation with Structural Correspondence Learning." In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pp. 120–128.

Buchholz, S. and Marsi, E. (2006). "CoNLL-X Shared Task on Multilingual Dependency Parsing." In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pp. 149–164.

Gildea, D. (2001). "Corpus Variation and Parser Performance." In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pp. 167–202.

Iida, R., Komachi, M., Inui, K., and Matsumoto, Y. (2007). "Annotating a Japanese Text Corpus with Predicate-argument and Coreference Relations." In *Proceedings of the Linguistic Annotation Workshop*, pp. 132–139.

Keene, D., Hatori, H., Yamada, H., and Irabu, S. (1992). *Japanese-English Sentence Equivalents (in Japanese)* (Electronic Book edition). Asahi Press.

Kudo, T. and Matsumoto, Y. (2000). "Japanese dependency structure analysis based on support vector machines." In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pp. 18–25.

Kudo, T. and Matsumoto, Y. (2002). "Japanese dependency analysis using cascaded chunking." In *Proceedings of the Sixth Conference on Computational Natural Language Learning*, Vol. 25, pp. 1–7.

Liang, P., Daumé III, H., and Klein, D. (2008). "Structure compilation: trading structure for features." In *Proceedings of the 25th International Conference on Machine Learning.*

Maekawa, K. (2008). "Balanced Corpus of Contemporary Written Japanese." In *Proceedings of the 6th Workshop on Asian Language Resources*, pp. 101–102.

McClosky, D., Charniak, E., and Johnson, M. (2006). "Reranking and Self-training for Parser Adaptation." In *Proceedings of the 44th Annual Meeting of the Association for Computational*

Linguistics, pp. 337–344.

McDonald, R. and Pereira, F. (2006). "Online Learning of Approximate Dependency Parsing Algorithms." In *Proceedings of the Eleventh European Chapter of the Association for Computational Linguistics*, Vol. 6.

McDonald, R., Pereira, F., Ribarov, K., and Hajič, J. (2005). "Non-projective Dependency Parsing Using Spanning Tree Algorithms." In *Proceedings of the 2005 Conference on Empirical Methods in Natural Language Processing*, pp. 523–530.

Miyao, Y., Sagae, K., Saetre, R., Matsuzaki, T., and Tsujii, J. (2009). "Evaluating Contributions of Natural Language Parsers to Protein-Protein Interaction Extraction." *Bioinformatics*, **25** (3).

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

Neubig, G. and Mori, S. (2010). "Word-based Partial Annotation for Efficient Corpus Construction." In *Proceedings of the Seventh International Conference on Language Resources and Evaluation.*

Neubig, G., Nakata, Y., and Mori, S. (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.*

Nivre, J., Hall, J., Kübler, S., McDonald, R., Nilsson, J., Riedel, S., and Yuret, D. (2007). "The CoNLL 2007 Shared Task on Dependency Parsing." In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007.*

Nivre, J., Hall, J., and Nilsson, J. (2006a). "MaltParser: A Data-driven Parser-generator for Dependency Parsing." In *Proceedings of the Fifth International Conference on Language Resources and Evaluation.*

Nivre, J., Hall, J., Nilsson, J., Eryiğit, G., and Marinov, S. (2006b). "Labeled Pseudo-projective Dependency Parsing with Support Vector Machines." In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pp. 221–225.

Nivre, J. and Scholz, M. (2004). "Deterministic Dependency Parsing of English Text." In *Proceedings of the 20th International Conference on Computational Linguistics.*

Osborne, M. and Baldridge, J. (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*, pp. 89–96.

Petrov, S., Chang, P.-C., Ringgaard, M., and Alshawi, H. (2010). "Uptraining for Accurate Deterministic Question Parsing." In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pp. 705–713.

Roark, B. and Bacchiani, M. (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*, pp. 126–133.

Sassano, M. (2005). "Using a Partially Annotated Corpus to Build a Dependency Parser for Japanese." In *Proceedings of the Second International Joint Conference on Natural Language Processing.*

Sassano, M. and Kurohashi, S. (2009). "A Unified Single Scan Algorithm for Japanese Base Phrase Chunking and Dependency Parsing." In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP.*

Sassano, M. and Kurohashi, S. (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*, pp. 356–365.

Spreyer, K. and Kuhn, J. (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)*, pp. 12–20.

Spreyer, K., Øvrelid, L., and Kuhn, J. (2010). "Training parsers on partial trees: a cross-language comparison." In *Proceedings of the Seventh International Conference on Language Resources and Evaluation.*

Suzuki, J., Isozaki, H., Carreras, X., and Collins, M. (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*, pp. 551–560.

Tang, M., Luo, X., and Roukos, S. (2002). "Active Learning for Statistical Natural Language Parsing." In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 120–127.

Tsuboi, Y., Kashima, H., Mori, S., Oda, H., and Matsumoto, Y. (2008). "Training Conditional Random Fields Using Incomplete Annotations." In *Proceedings of the 22th International Conference on Computational Linguistics.*

Uchimoto, K., Sekine, S., and Isahara, H. (1999). "Japanese dependency structure analysis based on maximum entropy models." In *Proceedings of the Ninth European Chapter of the Association for Computational Linguistics*, pp. 196–203.

Wang, Q. I., Lin, D., and Schuurmans, D. (2007). "Simple training of dependency parsers via structured boosting." In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence.*

Yamada, K. and Knight, K. (2001). "A Syntax-Based Statistical Machine Translation Model." In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics.*

Daniel Flannery: Daniel Flannery received his B.S. from Michigan State University in 2005. He is currently a masters student in the Graduate School of Informatics at Kyoto University. His research interests include dependency parsing and machine translation. He is a member of the Association for Natural Language Processing.

Yusuke Miyao: Yusuke Miyao received his BSc and MSc from the University of Tokyo in 1998 and in 2000 respectively, and his PhD from the University of Tokyo in 2006. He has been a research associate at the University of Tokyo from 2001, and an associate professor at the National Institute of Informatics from 2010. Member of Japan Association for Natural Language Processing, Information Processing Society of Japan, Japan Society for Artificial Intelligence, and Association for Computational Linguistics.

Graham Neubig: Graham Neubig received his B.S. from University of Illinois, Urbana-Champaign, U.S.A, in 2005, and his M.E. and Ph.D. in informatics from Kyoto University, Kyoto, Japan in 2010 and 2012 respectively. From 2012, he has served as an Assistant Professor at the Nara Institute of Science and Technology. His research interests include speech and natural language processing, with a focus on unsupervised learning for applications such as automatic speech recognition and machine translation.

Shinsuke Mori: Shinsuke Mori received his B.S., M.S., and Ph.D. in electrical engineering from Kyoto University, Kyoto, Japan in 1993, 1995, and 1998, respectively. After joining the Tokyo Research Laboratory of International Business Machines (IBM) in 1998, he studied language modeling and its application to speech recognition and language processing. He is currently an associate professor at the Academic Center for Computing and Media Studies, Kyoto University. His research interests include speech recognition and natural language processing. He is a member of Japan Association for Natural Language Processing, and the Information Processing Society of Japan.