Using Smaller Constituents Rather Than Sentences in Active Learning for Japanese Dependency Parsing

Manabu Sassano
Yahoo Japan Corporation
Midtown Tower,
9-7-1 Akasaka, Minato-ku,
Tokyo 107-6211, Japan
msassano@yahoo-corp.jp

Sadao Kurohashi
Graduate School of Informatics,
Kyoto University
Yoshida-honmachi, Sakyo-ku,
Kyoto 606-8501, Japan
kuro@i.kyoto-u.ac.jp

Abstract

We investigate active learning methods for Japanese dependency parsing. We propose active learning methods of using partial dependency relations in a given sentence for parsing and evaluate their effectiveness empirically. Furthermore, we utilize syntactic constraints of Japanese to obtain more labeled examples from precious labeled ones that annotators give. Experimental results show that our proposed methods improve considerably the learning curve of Japanese dependency parsing. In order to achieve an accuracy of over 88.3%, one of our methods requires only 34.4% of labeled examples as compared to passive learning.

1 Introduction

Reducing annotation cost is very important because supervised learning approaches, which have been successful in natural language processing, require typically a large number of labeled examples. Preparing many labeled examples is time consuming and labor intensive.

One of most promising approaches to this issue is active learning. Recently much attention has been paid to it in the field of natural language processing. Various tasks have been targeted in the research on active learning. They include word sense disambiguation, e.g., (Zhu and Hovy, 2007), POS tagging (Ringger et al., 2007), named entity recognition (Laws and Schütze, 2008), word segmentation, e.g., (Sassano, 2002), and parsing, e.g., (Tang et al., 2002; Hwa, 2004).

It is the main purpose of this study to propose methods of improving active learning for parsing by using a smaller constituent than a sentence as a unit that is selected at each iteration of active learning. Typically in active learning for parsing a sentence has been considered to be a basic unit for selection. Small constituents such as chunks have not been used in sample selection for parsing. We use Japanese dependency parsing as a target task in this study since a simple and efficient algorithm of parsing is proposed and, to our knowledge, active learning for Japanese dependency parsing has never been studied.

The remainder of the paper is organized as follows. Section 2 describes the basic framework of active learning which is employed in this research. Section 3 describes the syntactic characteristics of Japanese and the parsing algorithm that we use. Section 4 briefly reviews previous work on active learning for parsing and discusses several research challenges. In Section 5 we describe our proposed methods and others of active learning for Japanese dependency parsing. Section 6 describes experimental evaluation and discussion. Finally, in Section 7 we conclude this paper and point out some future directions.

2 Active Learning

2.1 Pool-based Active Learning

Our base framework of active learning is based on the algorithm of (Lewis and Gale, 1994), which is called *pool-based active learning*. Following their sequential sampling algorithm, we show in Figure 1 the basic flow of pool-based active learning. Various methods for selecting informative examples can be combined with this framework.

2.2 Selection Algorithm for Large Margin Classifiers

One of the most accurate approaches to classification tasks is an approach with large margin classifiers. Suppose that we are given data points $\{x_i\}$ such that the associated label y_i will be either -1 or 1, and we have a hyperplane of some large margin classifier defined by $\{x : f(x) = 0\}$ where the

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 356–365,
Uppsala, Sweden, 11-16 July 2010.

1. Build an initial classifier from an initial labeled training set.
2. While resources for labeling examples are available
   (a) Apply the current classifier to each unlabeled example
   (b) Find the m examples which are most *informative* for the classifier
   (c) Have annotators label the m examples
   (d) Train a new classifier on all labeled examples

Figure 1: Flow of the pool-based active learning

	Lisa-ga	kare-ni	ano	pen-wo	age-ta.
	Lisa-subj	to him	that	pen-acc	give-past.
ID	0	1	2	3	4
Head	4	4	3	4	-

Figure 2: Sample sentence. An English translation is "Lisa gave that pen to him."

classification function is $G(x) = \text{sign}\{f(x)\}$. In pool-based active learning with large margin classifiers, selection of examples can be done as follows:

1. Compute $f(x_i)$ over all unlabeled examples x_i in the pool.
2. Sort x_i with $|f(x_i)|$ in ascending order.
3. Select top m examples.

This type of selection methods with SVMs is discussed in (Tong and Koller, 2000; Schohn and Cohn, 2000). They obtain excellent results on text classification. These selection methods are simple but very effective.

3 Japanese Parsing

3.1 Syntactic Units

A basic syntactic unit used in Japanese parsing is a *bunsetsu*, the concept of which was initially introduced by Hashimoto (1934). We assume that in Japanese we have a sequence of *bunsetsu*s before parsing a sentence. A *bunsetsu* contains one or more content words and zero or more function words.

A sample sentence in Japanese is shown in Figure 2. This sentence consists of five bunsetsus: *Lisa-ga, kare-ni, ano, pen-wo,* and *age-ta* where *ga, ni,* and *wo* are postpositions and *ta* is a verb ending for past tense.

3.2 Constraints of Japanese Dependency Analysis

Japanese is a head final language and in written Japanese we usually hypothesize the following:

- Each bunsetsu has only one head except the rightmost one.
- Dependency links between bunsetsus go from left to right.
- Dependencies do not cross one another.

We can see that these constraints are satisfied in the sample sentence in Figure 2. In this paper we also assume that the above constraints hold true when we discuss algorithms of Japanese parsing and active learning for it.

3.3 Algorithm of Japanese Dependency Parsing

We use Sassano's algorithm (Sassano, 2004) for Japanese dependency parsing. The reason for this is that it is very accurate and efficient[1]. Furthermore, it is easy to implement. His algorithm is one of the simplest form of shift-reduce parsers and runs in linear-time.[2] Since Japanese is a head final language and its dependencies are projective as described in Section 3.2, that simplification can be made.

The basic flow of Sassano's algorithm is shown in Figure 3, which is slightly simplified from the original by Sassano (2004). When we use this algorithm with a machine learning-based classifier, function Dep() in Figure 3 uses the classifier to decide whether two bunsetsus have a dependency relation. In order to prepare training examples for the trainable classifier used with his algorithm, we first have to convert a treebank to suitable labeled instances by using the algorithm in Figure 4. Note

[1] Iwatate et al. (2008) compare their proposed algorithm with various ones that include Sassano's, cascaded chunking (Kudo and Matsumoto, 2002), and one in (McDonald et al., 2005). Kudo and Matsumoto (2002) compare cascaded chunking with the CYK method (Kudo and Matsumoto, 2000). After considering these results, we have concluded so far that Sassano's is a reasonable choice for our purpose.

[2] Roughly speaking, Sassano's is considered to be a simplified version, which is modified for head final languages, of Nivre's (Nivre, 2003). Classifiers with Nivre's are required to handle multiclass prediction, while binary classifiers can work with Sassano's for Japanese.

Input: w_i: bunsetsus in a given sentence.
N: the number of bunsetsus.
Output: h_j: the head IDs of bunsetsus w_j.
Functions: Push(i, s): pushes i on the stack s.
Pop(s): pops a value off the stack s.
Dep(j, i, w): returns true when w_j should modify w_i. Otherwise returns false.

```
procedure Analyze(w, N, h)
var s: a stack for IDs of modifier bunsetsus
begin
  {−1 indicates no modifier candidate}
  Push(−1, s);
  Push(0, s);
  for i ← 1 to N − 1 do begin
    j ← Pop(s);
    while (j ≠ −1
        and ((i = N − 1) or Dep(j, i, w)) ) do
    begin
      h_j ← i;
      j ← Pop(s)
    end
    Push(j, s);
    Push(i, s)
  end
end
```

Figure 3: Algorithm of Japanese dependency parsing

that the algorithm in Figure 4 does not generate every pair of bunsetsus.[3]

4 Active Learning for Parsing

Most of the methods of active learning for parsing in previous work use selection of sentences that seem to contribute to the improvement of accuracy (Tang et al., 2002; Hwa, 2004; Baldridge and Osborne, 2004). Although Hwa suggests that sample selection for parsing would be improved by selecting finer grained constituents rather than sentences (Hwa, 2004), such methods have not been investigated so far.

Typical methods of selecting sentences are

[3]We show a sample set of generated examples for training the classifier of the parser in Figure 3. By using the algorithm in Figure 4, we can obtain labeled examples from the sample sentences in Figure 2: $\{0, 1,$ "O"$\}$, $\{1, 2,$ "O"$\}$, $\{2, 3,$ "D"$\}$, and $\{1, 3,$ "O"$\}$. Please see Section 5.2 for the notation used here. For example, an actual labeled instance generated from $\{2, 3,$ "D"$\}$ will be like "label=D, features={modifier-content-word=ano, ..., head-content-word=pen, ...}."

Input: h_i: the head IDs of bunsetsus w_i.
Function: Dep(j, i, w, h): returns true if $h_j = i$.
Otherwise returns false. Also prints a feature vector with a label according to h_j.

```
procedure Generate(w, N, h)
begin
  Push(−1, s);
  Push(0, s);
  for i ← 1 to N − 1 do begin
    j ← Pop(s);
    while (j ≠ −1
      and ((i = N − 1) or Dep(j, i, w, h)) ) do
    begin
      j ← Pop(s)
    end
    Push(j, s);
    Push(i, s)
  end
end
```

Figure 4: Algorithm of generating training examples

based on some entropy-based measure of a given sentence (e.g., (Tang et al., 2002)). We cannot use this kind of measures when we want to select other smaller constituents than sentences. Other bigger problem is an algorithm of parsing itself. If we sample smaller units rather than sentences, we have partially annotated sentences and have to use a parsing algorithm that can be trained from incompletely annotated sentences. Therefore, it is difficult to use some of probabilistic models for parsing. [4]

5 Active Learning for Japanese Dependency Parsing

In this section we describe sample selection methods which we investigated.

5.1 Sentence-wise Sample Selection

Passive Selection (Passive) This method is to select sequentially sentences that appear in the training corpus. Since it gets harder for the readers to reproduce the same experimental setting, we

[4]We did not employ *query-by-committee* (QBC) (Seung et al., 1992), which is another important general framework of active learning, since the selection strategy with large margin classifiers (Section 2.2) is much simpler and seems more practical for active learning in Japanese dependency parsing with smaller constituents.

avoid to use random sampling in this paper.

Minimum Margin Selection (Min) This method is to select sentences that contain bunsetsu pairs which have smaller margin values of outputs of the classifier used in parsing. The procedure of selection of MIN are summarized as follows. Assume that we have sentences s_i in the pool of unlabeled sentences.

1. Parse s_i in the pool with the current model.
2. Sort s_i with $\min |f(x_k)|$ where x_k are bunsetsu pairs in the sentence s_i. Note that x_k are not all possible bunsetsu pairs in s_i and they are limited to bunsetsu pairs checked in the process of parsing s_i.
3. Select top m sentences.

Averaged Margin Selection (Avg) This method is to select sentences that have smaller values of averaged margin values of outputs of the classifier in a give sentences over the number of decisions which are carried out in parsing. The difference between AVG and MIN is that for AVG we use $\sum |f(x_k)|/l$ where l is the number of calling Dep() in Figure 3 for the sentence s_i instead of $\min |f(x_k)|$ for MIN.

5.2 Chunk-wise Sample Selection

In chunk-wise sample selection, we select bunsetsu pairs rather than sentences. Bunsetsu pairs are selected from different sentences in a pool. This means that structures of sentences in the pool are partially annotated.

Note that we do not use every bunsetsu pair in a sentence. When we use Sassano's algorithm, we have to generate training examples for the classifier by using the algorithm in Figure 4. In other words, we should not sample bunsetsu pairs independently from a given sentence.

Therefore, we select bunsetsu pairs that have smaller margin values of outputs given by the classifier during the parsing process. All the sentences in the pool are processed by the current parser. We cannot simply split the sentences in the pool into labeled and unlabeled ones because we do not select every bunsetsu pair in a given sentence.

Naive Selection (Naive) This method is to select bunsetsu pairs that have smaller margin values of outputs of the classifier. Then it is assumed that annotators would label either "D" for the two bunsetsu having a dependency relation or "O", which represents the two does not.

Modified Simple Selection (ModSimple) Although NAIVE seems to work well, it did not (discussed later). MODSIMPLE is to select bunsetsu pairs that have smaller margin values of outputs of the classifier, which is the same as in NAIVE. The difference between MODSIMPLE and NAIVE is the way annotators label examples. Assume that we have an annotator and the learner selects some bunsetsu pair of the j-th bunsetsu and the i-th bunsetsu such that $j < i$. The annotator is then asked what the head of the j-th bunsetsu is. We define here the head bunsetsu is the k-th one.

We differently generate labeled examples from the information annotators give according to the relation among bunsetsus j, i, and k.

Below we use the notation $\{s, t, \text{"D"}\}$ to denote that the s-th bunsetsu modifies the t-th one. The use of "O" instead of "D" indicates that the s-th does not modify the t-th. That is generating $\{s, t, \text{"D"}\}$ means outputting an example with the label "D".

Case 1 if $j < i < k$, then generate $\{j, i, \text{"O"}\}$ and $\{j, k, \text{"D"}\}$.

Case 2 if $j < i = k$, then generate $\{j, k, \text{"D"}\}$.

Case 3 if $j < k < i$, then generate $\{j, k, \text{"D"}\}$. Note that we do not generate $\{j, i, \text{"O"}\}$ in this case because in Sassano's algorithm we do not need such labeled examples if j depends on k such that $k < i$.

Syntactically Extended Selection (Syn) This selection method is one based on MODSIMPLE and extended to generate more labeled examples for the classifier. You may notice that more labeled examples for the classifier can be generated from a single label which the annotator gives. Syntactic constraints of the Japanese language allow us to extend labeled examples.

For example, suppose that we have four bunsetsus A, B, C, and D in this order. If A depends on C, i.e., the head of A is C, then it is automatically derived that B also should depend on C because the Japanese language has the no-crossing constraint for dependencies (Section 3.2). By utilizing this property we can obtain more labeled examples from a single labeled one annotators give. In the example above, we obtain $\{A, B, \text{"O"}\}$ and $\{B, C, \text{"D"}\}$ from $\{A, C, \text{"D"}\}$.

Although we can employ various extensions to MODSIMPLE, we use a rather simple extension in this research.

Case 1 if $(j < i < k)$, then generate

- $\{j, i, \text{“O”}\}$,
- $\{k-1, k, \text{“D”}\}$ if $k-1 > j$,
- and $\{j, k, \text{“D”}\}$.

Case 2 if $(j < i = k)$, then generate

- $\{k-1, k, \text{“D”}\}$ if $k-1 > j$,
- and $\{j, k, \text{“D”}\}$.

Case 3 if $(j < k < i)$, then generate

- $\{k-1, k, \text{“D”}\}$ if $k-1 > j$,
- and $\{j, k, \text{“D”}\}$.

In SYN as well as MODSIMPLE, we generate examples with ”O” only for bunsetsu pairs that occur to the left of the correct head (i.e., case 1).

6 Experimental Evaluation and Discussion

6.1 Corpus

In our experiments we used the Kyoto University Corpus Version 2 (Kurohashi and Nagao, 1998). Initial seed sentences and a pool of unlabeled sentences for training are taken from the articles on January 1st through 8th (7,958 sentences) and the test data is a set of sentences in the articles on January 9th (1,246 sentences). The articles on January 10th were used for development. The split of these articles for training/test/development is the same as in (Uchimoto et al., 1999).

6.2 Averaged Perceptron

We used the averaged perceptron (AP) (Freund and Schapire, 1999) with polynomial kernels. We set the degree of the kernels to 3 since cubic kernels with SVM have proved effective for Japanese dependency parsing (Kudo and Matsumoto, 2000; Kudo and Matsumoto, 2002). We found the best value of the epoch T of the averaged perceptron by using the development set. We fixed $T = 12$ through all experiments for simplicity.

6.3 Features

There are features that have been commonly used for Japanese dependency parsing among related papers, e.g., (Kudo and Matsumoto, 2002; Sassano, 2004; Iwatate et al., 2008). We also used the same features here. They are divided into three groups: modifier *bunsetsu* features, head *bunsetsu* features, and gap features. A summary of the features is described in Table 1.

6.4 Implementation

We implemented a parser and a tool for the averaged perceptron in C++ and used them for experiments. We wrote the main program of active learning and some additional scripts in Perl and sh.

6.5 Settings of Active Learning

For initial seed sentences, first 500 sentences are taken from the articles on January 1st. In experiments about sentence wise selection, 500 sentences are selected at each iteration of active learning and labeled[5] and added into the training data. In experiments about chunk wise selection 4000 pairs of bunsetsus, which are roughly equal to the averaged number of bunsetsus in 500 sentences, are selected at each iteration of active learning.

6.6 Dependency Accuracy

We use dependency accuracy as a performance measure of a parser. The dependency accuracy is the percentage of correct dependencies. This measure is commonly used for the Kyoto University Corpus.

6.7 Results and Discussion

Learning Curves First we compare methods for sentence wise selection. Figure 5 shows that MIN is the best among them, while AVG is not good and similar to PASSIVE. It is observed that active learning with large margin classifiers also works well for Sassano’s algorithm of Japanese dependency parsing.

Next we compare chunk-wise selection with sentence-wise one. The comparison is shown in Figure 6. Note that we must carefully consider how to count labeled examples. In sentence wise selection we obviously count the number of sentences. However, it is impossible to count such number when we label bunsetsus pairs.

Therefore, we use the number of bunsetsus that have an annotated head. Although we know this may not be a completely fair comparison, we believe our choice in this experiment is reasonable

[5] In our experiments human annotators do not give labels. Instead, labels are given virtually from correct ones that the Kyoto University Corpus has.

Bunsetsu features for modifiers and heads	rightmost content word, rightmost function word, punctuation, parentheses, location (BOS or EOS)
Gap features	distance (1, 2–5, or 6 $\leq$), particles, parentheses, punctuation

Table 1: Features for deciding a dependency relation between two bunsetsus. Morphological features for each word (morpheme) are major part-of-speech (POS), minor POS, conjugation type, conjugation form, and surface form.

for assessing the effect of reduction by chunk-wise selection.

In Figure 6 NAIVE has a better learning curve compared to MIN at the early stage of learning. However, the curve of NAIVE declines at the later stage and gets worse than PASSIVE and MIN.

Why does this phenomenon occur? It is because each bunsetsu pair is not independent and pairs in the same sentence are related to each other. They satisfy the constraints discussed in Section 3.2. Furthermore, the algorithm we use, i.e., Sassano's, assumes these constraints and has the specific order for processing bunsetsu pairs as we see in Figure 3. Let us consider the meaning of $\{j, i, \text{“O”}\}$ if the head of the j-th bunsetsu is the k-th one such that $j < k < i$. In the context of the algorithm in Figure 3, $\{j, i, \text{“O”}\}$ actually means that the j-th bunsetsu modifies th l-th one such that $i < l$. That is “O” does not simply mean that two bunsetsus does not have a dependency relation. Therefore, we should not generate $\{j, i, \text{“O”}\}$ in the case of $j < k < i$. Such labeled instances are not needed and the algorithm in Figure 4 does not generate them even if a fully annotated sentence is given. Based on the analysis above, we modified NAIVE and defined MODSIMPLE, where unnecessary labeled examples are not generated.

Now let us compare NAIVE with MODSIMPLE (Figure 7). MODSIMPLE is almost always better than PASSIVE and does not cause a significant deterioration of accuracy unlike NAIVE.[6]

Comparison of MODSIMPLE and SYN is shown in Figure 8. Both exhibit a similar curve. Figure 9 shows the same comparison in terms of required queries to human annotators. It shows that SYN is better than MODSIMPLE especially at the earlier stage of active learning.

Reduction of Annotations Next we examined the number of labeled bunsetsus to be required in

[6]We have to carefully see the curves of NAIVE and MODSIMPLE. In Figure 7 at the early stage NAIVE is slightly better than MODSIMPLE, while in Figure 9 NAIVE does not outperform MODSIMPLE. This is due to the difference of the way of accessing annotation efforts.

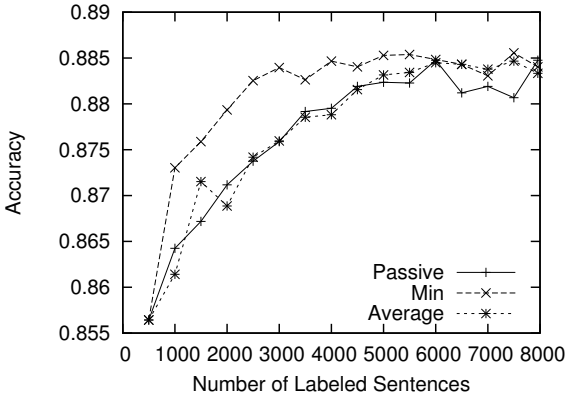

Figure 5: Learning curves of methods for sentence wise selection

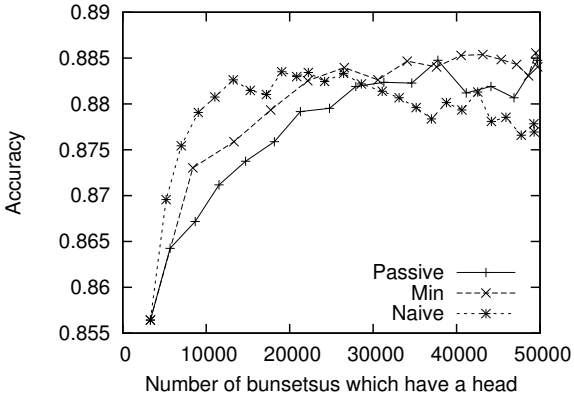

Figure 6: Learning curves of MIN (sentence-wise) and NAIVE (chunk-wise).

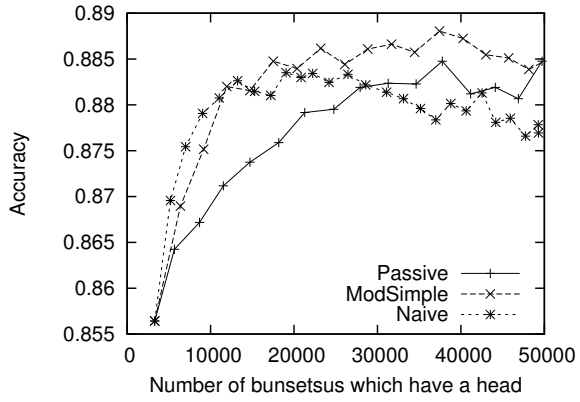

Figure 7: Learning curves of NAIVE, MODSIMPLE and PASSIVE in terms of the number of bunsetsus that have a head.

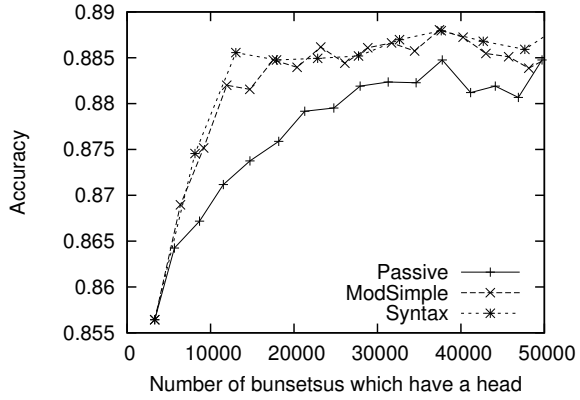

Figure 8: Learning curves of MODSIMPLE and SYN in terms of the number of bunsetsus which have a head.

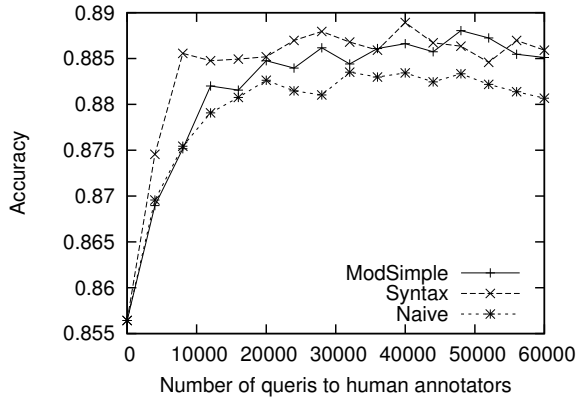

Figure 9: Comparison of MODSIMPLE and SYN in terms of the number of queries to human annotators

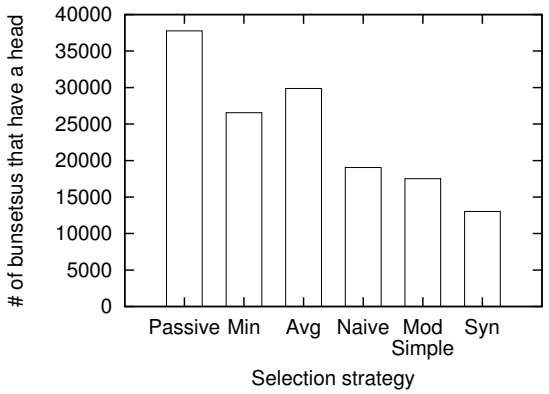

Figure 10: Number of labeled bunsetsus to be required to achieve an accuracy of over 88.3%.

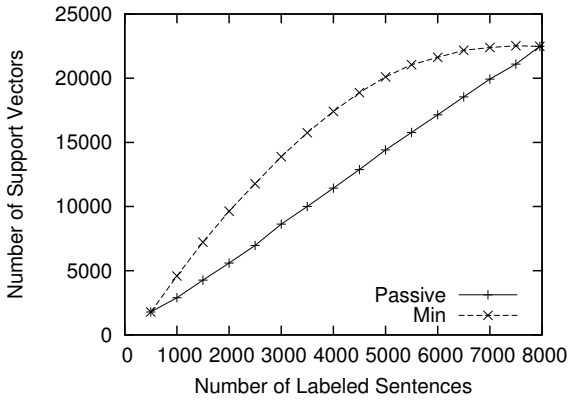

Figure 11: Changes of number of support vectors in sentence-wise active learning

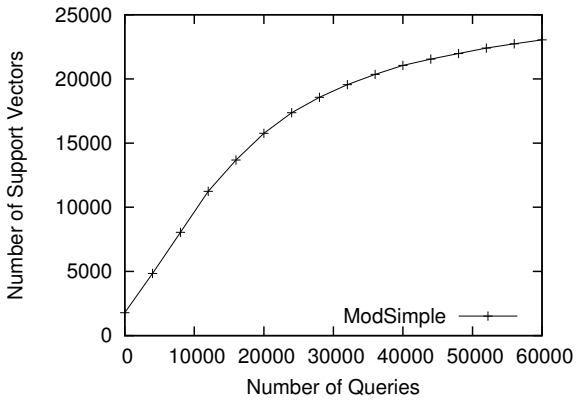

Figure 12: Changes of number of support vectors in chunk-wise active learning (MODSIMPLE)

order to achieve a certain level of accuracy. Figure 10 shows that the number of labeled bunsetsus to achieve an accuracy of over 88.3% depending on the active learning methods discussed in this research.

PASSIVE needs 37766 labeled bunsetsus which have a head to achieve an accuracy of 88.48%, while SYN needs 13021 labeled bunsetsus to achieve an accuracy of 88.56%. SYN requires only 34.4% of the labeled bunsetsu pairs that PASSIVE requires.

Stopping Criteria It is known that increment rate of the number of support vectors in SVM indicates saturation of accuracy improvement during iterations of active learning (Schohn and Cohn, 2000). It is interesting to examine whether the observation for SVM is also useful for support vectors[7] of the averaged perceptron. We plotted changes of the number of support vectors in the cases of both PASSIVE and MIN in Figure 11 and changes of the number of support vectors in the case of MODSIMPLE in Figure 12. We observed that the increment rate of support vectors mildly gets smaller. However, it is not so clear as in the case of text classification in (Schohn and Cohn, 2000).

Issues on Accessing the Total Cost of Annotation In this paper, we assume that each annotation cost for dependency relations is constant. It is however not true in an actual annotation work.[8] In addition, we have to note that it may be easier to annotate a whole sentence than some bunsetsu pairs in a sentence[9]. In a real annotation task, it will be better to show a whole sentence to annotators even when annotating some part of the sentence.

Nevertheless, it is noteworthy that our research shows the minimum number of annotations in preparing training examples for Japanese dependency parsing. The methods we have proposed must be helpful when checking repeatedly annotations that are important and might be wrong or difficult to label while building an annotated corpus. They also will be useful for domain adaptation of a dependency parser.[10]

Applicability to Other Languages and Other Parsing Algorithms We discuss here whether or not the proposed methods and the experiments are useful for other languages and other parsing algorithms. First we take languages similar to Japanese in terms of syntax, i.e., Korean and Mongolian. These two languages are basically head-final languages and have similar constraints in Section 3.2. Although no one has reported application of (Sassano, 2004) to the languages so far, we believe that similar parsing algorithms will be applicable to them and the discussion in this study would be useful.

On the other hand, the algorithm of (Sassano, 2004) cannot be applied to head-initial languages such as English. If target languages are assumed to be projective, the algorithm of (Nivre, 2003) can be used. It is highly likely that we will invent the effective use of finer-grained constituents, e.g., head-modifier pairs, rather than sentences in active learning for Nivre's algorithm with large margin classifiers since Sassano's seems to be a simplified version of Nivre's and they have several properties in common. However, syntactic constraints in European languages like English may be less helpful than those in Japanese because their dependency links do not have a single direction.

Even though the use of syntactic constraints is limited, smaller constituents will still be useful for other parsing algorithms that use some deterministic methods with machine learning-based classifiers. There are many algorithms that have such a framework, which include (Yamada and Matsumoto, 2003) for English and (Kudo and Matsumoto, 2002; Iwatate et al., 2008) for Japanese. Therefore, effective use of smaller constituents in active learning would not be limited to the specific algorithm.

7 Conclusion

We have investigated that active learning methods for Japanese dependency parsing. It is observed that active learning of parsing with the averaged perceptron, which is one of the large margin classifiers, works also well for Japanese dependency analysis.

[7] Following (Freund and Schapire, 1999), we use the term "support vectors" for AP as well as SVM. "Support vectors" of AP means vectors which are selected in the training phase and contribute to the prediction.

[8] Thus it is very important to construct models for estimating the actual annotation cost as Haertel et al. (2008) do.

[9] Hwa (2004) discusses similar aspects of researches on active learning.

[10] Ohtake (2006) examines heuristic methods of selecting sentences.

In addition, as far as we know, we are the first to propose the active learning methods of using partial dependency relations in a given sentence for parsing and we have evaluated the effectiveness of our methods. Furthermore, we have tried to obtain more labeled examples from precious labeled ones that annotators give by utilizing syntactic constraints of the Japanese language. It is noteworthy that linguistic constraints have been shown useful for reducing annotations in active learning for NLP.

Experimental results show that our proposed methods have improved considerably the learning curve of Japanese dependency parsing.

We are currently building a new annotated corpus with an annotation tool. We have a plan to incorporate our proposed methods to the annotation tool. We will use it to accelerate building of the large annotated corpus to improved our Japanese parser.

It would be interesting to explore the use of partially labeled constituents in a sentence in another language, e.g., English, for active learning.

Acknowledgements

We would like to thank the anonymous reviewers and Tomohide Shibata for their valuable comments.

References

Jason Baldridge and Miles Osborne. 2004. Active learning and the total cost of annotation. In *Proc. of EMNLP 2004*, pages 9–16.

Yoav Freund and Robert E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

Robbie Haertel, Eric Ringger, Kevin Seppi, James Carroll, and Peter McClanahan. 2008. Assessing the costs of sampling methods in active learning for annotation. In *Proc. of ACL-08: HLT, short papers (Companion Volume)*, pages 65–68.

Shinkichi Hashimoto. 1934. *Essentials of Japanese Grammar (Kokugoho Yousetsu) (in Japanese)*.

Rebecca Hwa. 2004. Sample selection for statistical parsing. *Computational Linguistics*, 30(3):253–276.

Masakazu Iwatate, Masayuki Asahara, and Yuji Matsumoto. 2008. Japanese dependency parsing using a tournament model. In *Proc. of COLING 2008*, pages 361–368.

Taku Kudo and Yuji Matsumoto. 2000. Japanese dependency structure analysis based on support vector machines. In *Proc. of EMNLP/VLC 2000*, pages 18–25.

Taku Kudo and Yuji Matsumoto. 2002. Japanese dependency analysis using cascaded chunking. In *Proc. of CoNLL-2002*, pages 63–69.

Sadao Kurohashi and Makoto Nagao. 1998. Building a Japanese parsed corpus while improving the parsing system. In *Proc. of LREC-1998*, pages 719–724.

Florian Laws and Hinrich Schütze. 2008. Stopping criteria for active learning of named entity recognition. In *Proc. of COLING 2008*, pages 465–472.

David D. Lewis and William A. Gale. 1994. A sequential algorithm for training text classifiers. In *Proc. of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proc. of ACL-2005*, pages 523–530.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proc. of IWPT-03*, pages 149–160.

Kiyonori Ohtake. 2006. Analysis of selective strategies to build a dependency-analyzed corpus. In *Proc. of COLING/ACL 2006 Main Conf. Poster Sessions*, pages 635–642.

Eric Ringger, Peter McClanahan, Robbie Haertel, George Busby, Marc Carmen, James Carroll, Kevin Seppi, and Deryle Lonsdale. 2007. Active learning for part-of-speech tagging: Accelerating corpus annotation. In *Proc. of the Linguistic Annotation Workshop*, pages 101–108.

Manabu Sassano. 2002. An empirical study of active learning with support vector machines for Japanese word segmentation. In *Proc. of ACL-2002*, pages 505–512.

Manabu Sassano. 2004. Linear-time dependency analysis for Japanese. In *Proc. of COLING 2004*, pages 8–14.

Greg Schohn and David Cohn. 2000. Less is more: Active learning with support vector machines. In *Proc. of ICML-2000*, pages 839–846.

H. S. Seung, M. Opper, and H. Sompolinsky. 1992. Query by committee. In *Proc. of COLT '92*, pages 287–294.

Min Tang, Xaoqiang Luo, and Salim Roukos. 2002. Active learning for statistical natural language parsing. In *Proc. of ACL-2002*, pages 120–127.

Simon Tong and Daphne Koller. 2000. Support vector machine active learning with applications to text classification. In *Proc. of ICML-2000*, pages 999–1006.

Kiyotaka Uchimoto, Satoshi Sekine, and Hitoshi Isahara. 1999. Japanese dependency structure analysis based on maximum entropy models. In *Proc. of EACL-99*, pages 196–203.

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

Jingbo Zhu and Eduard Hovy. 2007. Active learning for word sense disambiguation with methods for addressing the class imbalance problem. In *Proc. of EMNLP-CoNLL 2007*, pages 783–790.