Japanese Dependency Analysis using Cascaded Chunking

Taku Kudo and **Yuji Matsumoto**
Graduate School of Information Science,
Nara Institute of Science and Technology
{taku-ku,matsu}@is.aist-nara.ac.jp

Abstract

In this paper, we propose a new statistical Japanese dependency parser using a cascaded chunking model. Conventional Japanese statistical dependency parsers are mainly based on a probabilistic model, which is not always efficient or scalable. We propose a new method that is simple and efficient, since it parses a sentence deterministically only deciding whether the current segment modifies the segment on its immediate right hand side. Experiments using the Kyoto University Corpus show that the method outperforms previous systems as well as improves the parsing and training efficiency.

1 Introduction

Dependency analysis has been recognized as a basic process in Japanese sentence analysis, and a number of studies have been proposed. Japanese dependency structure is usually defined in terms of the relationship between phrasal units called *bunsetsu* segments (hereafter "segments").

Most of the previous statistical approaches for Japanese dependency analysis (Fujio and Matsumoto, 1998; Haruno et al., 1999; Uchimoto et al., 1999; Kanayama et al., 2000; Uchimoto et al., 2000; Kudo and Matsumoto, 2000) are based on a probabilistic model consisting of the following two steps. First, they estimate modification probabilities, in other words, how probable one segment tends to modify another. Second the optimal combination of dependencies is searched from the all candidates dependencies. Such a probabilistic model is not always efficient since it needs to calculate the probabilities for all possible dependencies and creates $n(n-1)/2$ (where n is the number of segments in a sentence) training examples per sentence. In addition, the probabilistic model assumes that each pairs of dependency structure is independent.

In this paper, we propose a new Japanese dependency parser which is more efficient and simpler than the probabilistic model, yet performs better in training and testing on the Kyoto University Corpus. The method parses a sentence deterministically only deciding whether the current segment modifies segment on its immediate right hand side. Moreover, it does not assume the independence constraint between dependencies

2 A Probabilistic Model

This section describes the general formulation of the probabilistic model for parsing which has been applied to Japanese statistical dependency analysis.

First of all, we define a sentence as a sequence of segments $B = \langle b_1, b_2 \ldots, b_m \rangle$ and its syntactic structure as a sequence of dependency patterns $D = \langle Dep(1), Dep(2), \ldots, Dep(m-1) \rangle$, where $Dep(i) = j$ means that the segment b_i depends on (modifies) segment b_j. In this framework, we assume that the dependency sequence D satisfies the following two constraints.

1. Japanese is a head-final language. Thus, except for the rightmost one, each segment modifies exactly one segment among the segments appearing to its right.
2. Dependencies do not cross one another.

Statistical dependency analysis is defined as a searching problem for the dependency pattern D that maximizes the conditional probability $P(D|B)$ of the input sequence under the above-mentioned constraints. If we assume that the dependency probabilities are mutually independent, $P(D|B)$ can be rewritten as:

$$P(D|B) = \prod_{i=1}^{m-1} P(Dep(i)=j \mid \mathbf{f}_{ij})$$
$$\mathbf{f}_{ij} = (f_1, \ldots, f_n) \in \mathrm{R}^n.$$

$P(Dep(i) = j \mid \mathbf{f}_{ij})$ represents the probability that b_i modifies b_j. $\mathbf{f}_{ij}$ is an n dimensional feature vector that represents various kinds of linguistic features related to the segments b_i and b_j.

We obtain $D_{best} = \mathrm{argmax}_D P(D|B)$ taking into all the combination of these probabilities. Generally, the optimal solution D_{best} can be identified by using bottom-up parsing algorithm such as CYK algorithm.

The problem in the dependency structure analysis is how to estimate the dependency probabilities accurately. A number of statistical and machine learning approaches, such as Maximum Likelihood estimation (Fujio and Matsumoto, 1998), Decision Trees (Haruno et al., 1999), Maximum Entropy models (Uchimoto et al., 1999; Uchimoto et al., 2000; Kanayama et al., 2000), and Support Vector Machines (Kudo and Matsumoto, 2000), have been applied to estimate these probabilities.

In order to apply a machine learning algorithm to dependency analysis, we have to prepare the positive and negative examples. Usually, in a probabilistic model, all possible pairs of segments that are in a dependency relation are used as positive examples, and two segments that appear in a sentence but are not in a dependency relation are used as negative examples. Thus, a total of $n(n-1)/2$ training examples (where n is the number of segments in a sentence) must be produced per sentence.

3 Cascaded Chunking Model

In the probabilistic model, we have to estimate the probabilities of each dependency relation. However, some machine learning algorithms, such as SVMs, cannot estimate these probabilities directly. Kudo and Matsumoto (2000) used the sigmoid function to obtain pseudo probabilities in SVMs. However, there is no theoretical endorsement for this heuristics.

Moreover, the probabilistic model is not good in its scalability since it usually requires a total of $n(n-1)/2$ training examples per sentence. It will be hard to combine the probabilistic model with some machine learning algorithms, such as SVMs, which require a polynomial computational cost on the number of given training examples.

In this paper, we introduce a new method for Japanese dependency analysis, which does not require the probabilities of dependencies and parses a sentence deterministically. The proposed method can be combined with any type of machine learning algorithm that has classification ability.

The original idea of our method stems from the cascaded chucking method which has been applied in English parsing (Abney, 1991). Let us introduce the basic framework of the cascaded chunking parsing method:

1. A sequence of base phrases is the input for this algorithm.
2. Scanning from the beginning of the input sentence, chunk a series of base phrases into a single non-terminal node.
3. For each chunked phrase, leave only the head phrase, and delete all the other phrases inside the chunk
4. Finish the algorithm if a single non-terminal node remains, otherwise return to the step 2 and repeat.

We apply this cascaded chunking parsing technique to Japanese dependency analysis. Since Japanese is a head-final language, and the chunking can be regarded as the creation of a dependency between two segments, we can simplify the process of Japanese dependency analysis as follows:

1. Put an **O** tag on all segments. The **O** tag indicates that the dependency relation of the current segment is undecided.
2. For each segment with an **O** tag, decide whether it modifies the segment on its immediate right hand side. If so, the **O** tag is replaced with a **D** tag.
3. Delete all segments with a D tag that are immediately followed by a segment with an O tag.
4. Terminate the algorithm if a single segment remains, otherwise return to step 2 and repeat.

Figure 1 shows an example of the parsing process with the cascaded chunking model.

The input for the model is the linguistic features related to the modifier and modifiee, and the output from the model is either of the tags (**D** or **O**). In training, the model simulates the parsing algorithm by consulting the correct answer from the training annotated corpus. During the training, positive (D) and negative (O) examples are collected. In testing, the model consults the trained system and parses the input with the cascaded chunking algorithm.

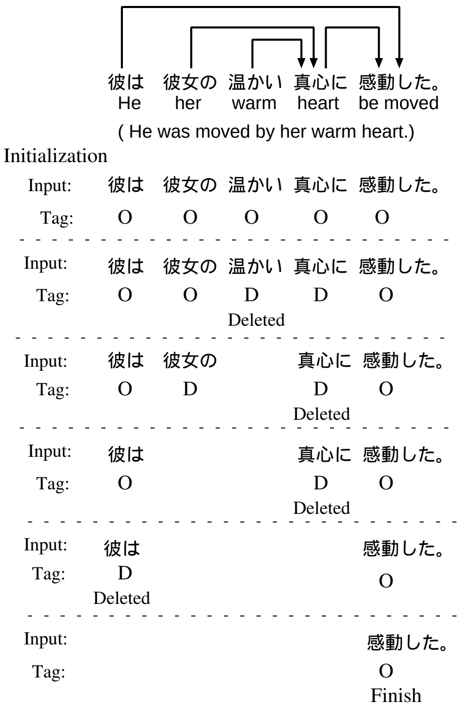

Figure 1: Example of the parsing process with cascaded chunking model

We think this proposed cascaded chunking model has the following advantages compared with the traditional probabilistic models.

- **Simple and Efficient**
  If we use the CYK algorithm, the probabilistic model requires $O(n^3)$ parsing time, (where n is the number of segments in a sentence.). On the other hand, the cascaded chunking model requires $O(n^2)$ in the worst case when all segments modify the rightmost segment. The actual parsing time is usually lower than $O(n^2)$, since most of segments modify segment on its immediate right hand side.

  Furthermore, in the cascaded chunking model, the training examples are extracted using the parsing algorithm itself. The training examples required for the cascaded chunking model is much smaller than that for the probabilistic model. The model reduces the training cost significantly and enables training using larger amounts of annotated corpus.

- **No assumption on the independence between dependency relations**
  The probabilistic model assumes that dependency relations are independent. However, there are some cases in which one cannot parse a sentence correctly with this assumption. For example, coordinate structures cannot be always parsed with the independence constraint. The cascaded chunking model parses and estimates relations simultaneously. This means that one can use all dependency relations, which have narrower scope than that of the current focusing relation being considered, as feature sets. We describe the details in the next section.

- **Independence from machine learning algorithm**
  The cascaded chunking model can be combined with any machine learning algorithm that works as a binary classifier, since the cascaded chunking model parses a sentence deterministically only deciding whether or not the current segment modifies the segment on its immediate right hand side. Probabilities of dependencies are not always necessary for the cascaded chunking model.

3.1 Dynamic and Static Features

Linguistic features that are supposed to be effective in Japanese dependency analysis are: head words and their parts-of-speech tags, functional words and inflection forms of the words that appear at the end of segments, distance between two segments, existence of punctuation marks. As those are solely defined by the pair of segments, we refer to them as the **static features**.

Japanese dependency relations are heavily constrained by such static features since the inflection forms and postpositional particles constrain the dependency relation. However, when a sentence is long and there are more than one possible dependency, static features, by themselves cannot determine the correct dependency.

To cope with this problem, Kudo and Matsumoto (2000) introduced a new type of features called **dynamic features**, which are created dynamically during the parsing process. For example, if some relation is determined, this modification relation may have some influence on other dependency relation. Therefore, once a segment has been determined to modify another segment, such information is kept in both of the segments and is added to them as a new feature. Specifically, we take the following three types of dynamic features in our experiments.

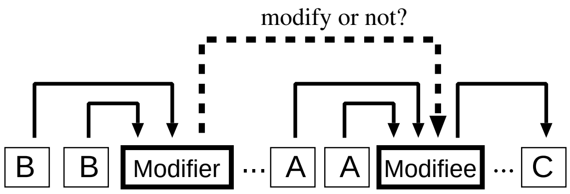

Figure 2: Three types of Dynamic Features

A. The segments which modify the current candidate modifiee. (boxes marked with A in Figure 2)

B. The segments which modify the current candidate modifier. (boxes marked with B in Figure 2)

C. The segment which is modified by the current candidate modifiee. (boxes marked with C in Figure 2)

4 Support Vector Machines

Although any kind of machine learning algorithm can be applied to the cascaded chunking model, we use Support Vector Machines (Vapnik, 1998) for our experiments because of their state-of-the-art performance and generalization ability.

SVM is a binary linear classifier trained from the samples, each of which belongs either to positive or negative class as follows: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)$ $(\mathbf{x}_i \in \mathbf{R}^n,\ y_i \in \{+1, -1\})$, where $\mathbf{x}_i$ is a feature vector of the i-th sample represented by an n dimensional vector, and y_i is the class (positive($+1$) or negative(-1) class) label of the i-th sample. SVMs find the optimal separating hyperplane $(\mathbf{w} \cdot \mathbf{x} + b)$ based on the maximal margin strategy. The margin can be seen as the distance between the critical examples and the separating hyperplane. We omit the details here, the maximal margin strategy can be realized by the following optimization problem:

$$\begin{array}{ll} \text{Minimize :} & L(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{Subject to :} & y_i[(\mathbf{w} \cdot \mathbf{x}_i) + b] \geq 1 \ (i = 1, \ldots, l). \end{array}$$

Furthermore, SVMs have the potential to carry out non-linear classifications. Though we leave the details to (Vapnik, 1998), the optimization problem can be rewritten into a dual form, where all feature vectors appear as their dot products. By simply substituting every dot product of $\mathbf{x}_i$ and $\mathbf{x}_j$ in dual form with a Kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$, SVMs can handle non-linear hypotheses. Among many kinds of Kernel functions available, we will focus on the d-th polynomial kernel: $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^d$. Use of d-th polynomial kernel functions allows us to build an optimal separating hyperplane which takes into account all combinations of features up to d.

5 Experiments and Discussion

5.1 Experimental Setting

We used the following two annotated corpora for our experiments.

- Standard data set
  This data set consists of the Kyoto University text corpus Version 2.0 (Kurohashi and Nagao, 1997). We used 7,958 sentences from the articles on January 1st to January 7th as training examples, and 1,246 sentences from the articles on January 9th as the test data. This data set was used in (Uchimoto et al., 1999; Uchimoto et al., 2000) and (Kudo and Matsumoto, 2000).

- Large data set
  In order to investigate the scalability of the cascaded chunking model, we prepared larger data set. We used all 38,383 sentences of the Kyoto University text corpus Version 3.0. The training and test data were generated by a two-fold cross validation.

The feature sets used in our experiments are shown in Table 1. The static features are basically taken from Uchimoto's list (Uchimoto et al., 1999).

Head Word (HW) is the rightmost content word in the segment. *Functional Word (FW)* is set as follows:

- *FW* = the rightmost functional word, if there is a functional word in the segment

- *FW* = the rightmost inflection form, if there is a predicate in the segment

- *FW* = same as the *HW*, otherwise.

The static features include the information on existence of brackets, question marks and punctuation marks, etc. Besides, there are features that show the relative relation of two segments, such as distance, and existence of brackets, quotation marks and punctuation marks between them.

For a segment X and its dynamic feature Y (where Y is of type A or B), we set the *Functional*

Representation (FR) feature of X based on the *FW* of X *(X-FW)* as follows:

- *FR* = lexical form of *X-FW* if POS of *X-FW* is particle, adverb, adnominal or conjunction
- *FR* = inflectional form of *X-FW* if *X-FW* has an inflectional form.
- *FR* = the POS tag of *X-FW*, otherwise.

For a segment X and its dynamic feature C, we set POS tag and POS-subcategory of the *HW* of X.

All our experiments are carried out on AlphaSever 8400 (21164A 500Mhz) for training and Linux (PentiumIII 1GHz) for testing. We used a third degree polynomial kernel function, which is exactly the same setting in (Kudo and Matsumoto, 2000).

Performance on the test data is measured using dependency accuracy and sentence accuracy. Dependency accuracy is the percentage of correct dependencies out of all dependency relations. Sentence accuracy is the percentage of sentences in which all dependencies are determined correctly.

5.2 Experimental Results

The results for the new cascaded chunking model as well as for the previous probabilistic model based on SVMs (Kudo and Matsumoto, 2000) are summarized in Table 2. We cannot employ the experiments for the probabilistic model using large dataset, since the data size is too large for our current SVMs learning program to terminate in a realistic time period.

Even though the number of training examples used for the cascaded chunking model is less than a quarter of that for the probabilistic model, and the used feature set is the same, dependency accuracy and sentence accuracy are improved using the cascaded chunking model (89.09% $\rightarrow$ 89.29%, 46.17% $\rightarrow$ 47.53%).

The time required for training and parsing are significantly reduced by applying the cascaded chunking model (336h.$\rightarrow$8h, 2.1sec.$\rightarrow$ 0.5sec.).

5.3 Probabilistic model vs. Cascaded Chunking model

As can be seen Table 2, the cascaded chunking model is more accurate, efficient and scalable than the probabilistic model. It is difficult to apply the probabilistic model to the large data set, since it takes no less than 336 hours (2 weeks) to carry out the experiments even with the standard data set, and SVMs require quadratic or more computational cost on the number of training examples.

For the first impression, it may seems natural that higher accuracy is achieved with the probabilistic model, since all candidate dependency relations are used as training examples. However, the experimental results show that the cascaded chunking model performs better. Here we list what the most significant contributions are and how well the cascaded chunking model behaves compared with the probabilistic model.

The probabilistic model is trained with all candidate pairs of segments in the training corpus. The problem of this training is that exceptional dependency relations may be used as training examples. For example, suppose a segment which appears to right hand side of the correct modifiee and has a similar content word, the pair with this segment becomes a negative example. However, this is negative because there is a better and correct candidate at a different point in the sentence. Therefore, this may not be a true negative example, meaning that this can be positive in other sentences. In addition, if a segment is not modified by a modifier because of cross dependency constraints but has a similar content word with correct modifiee, this relation also becomes an exception. Actually, we cannot ignore these exceptions, since most segments modify a segment on its immediate right hand side. By using all candidates of dependency relation as the training examples, we have committed to a number of exceptions which are hard to be trained upon. Looking in particular on a powerful heuristics for dependency structure analysis: “A segment tends to modify a nearer segment if possible,” it will be most important to train whether the current segment modifies the segment on its immediate right hand side. The cascaded chunking model is designed along with this heuristics and can remove the exceptional relations which has less potential to improve performance.

5.4 Effects of Dynamic Features

Figure 3 shows the relationship between the size of the training data and the parsing accuracy. This figure also shows the accuracy with and without the dynamic features. Generally, the results with the dynamic feature set is better than the results without it. The dynamic features constantly outperform static features when the size of the training data is large. In most cases, the improvements is considerable.

Table 3 summarizes the performance without some dynamic features. From these results, we can

Static Features	Modifier/Modifiee segments	*Head Word* (surface-form, POS, POS-subcategory, inflection-type, inflection-form), *Functional Word* (surface-form, POS, POS-subcategory, inflection-type, inflection-form), brackets, quotation-marks, punctuation-marks, position in sentence (beginning, end)
	Between two segments	distance(1,2-5,6-), case-particles, brackets, quotation-marks, punctuation-marks
Dynamic Features	Type A,B	Form of inflection represented with *Functional Representation*
	Type C	POS and POS-subcategory of Head word

Table 1: Features used in our experiments

Data Set	Standard		Large	
Model	Cascaded Chunking	Probabilistic	Cascaded Chunking	Probabilistic
Dependency Acc. (%)	89.29	89.09	90.46	N/A
Sentence Acc. (%)	47.53	46.17	53.16	N/A
# of training sentences	7,956	7,956	19,191	19,191
# of training examples	110,355	459,105	261,254	1,074,316
Training Time (hours)	8	336	48	N/A
Parsing Time (sec./sentence)	0.5	2.1	0.7	N/A

Table 2: Cascaded Chunking model vs Probabilistic model

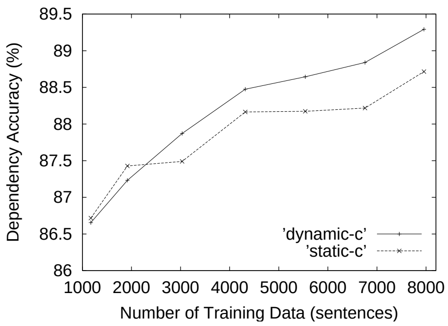

Figure 3: Training Data vs. Accuracy (cascaded chunking/standard data set)

Deleted type of dynamic feature	Diff. from the model with dynamic features	
	Dependency Acc.	Sentence Acc.
A	-0.28%	-0.89%
B	-0.10%	-0.89%
C	-0.28%	-0.56%
AB	-0.33%	-1.21%
AC	-0.55%	-0.97%
BC	-0.54%	-1.61%
ABC	-0.58%	-2.34%

Table 3: Effects of dynamic features with the standard data set

conclude that all dynamic features are effective in improving the performance.

5.5 Comparison with Related Work

Table 4 summarizes recent results on Japanese dependency analysis.

Uchimoto et al. (2000) report that using the Kyoto University Corpus for their training and testing, they achieve around 87.93% accuracy by building statistical model based on the Maximum Entropy framework. They extend the original probabilistic model, which learns only two class; ‘modify‘ and ‘not modify‘, to the one that learns three classes; ‘between‘, ‘modify‘ and ‘beyond‘. Their model can also avoid the influence of the exceptional dependency relations. Using same training and test data, we can achieve accuracy of 89.29%. The difference is considerable.

	Model	Training Corpus (# of sentences)	Acc. (%)
Our Model	Cascaded Chunking (SVMs)	Kyoto Univ. (19,191)	90.46
		Kyoto Univ. (7,956)	89.29
Kudo et al 00	Probabilistic model (SVMs)	Kyoto Univ. (7,956)	89.09
Uchimoto et al 00,98	Probabilistic model (ME + posterior context)	Kyoto Univ. (7,956)	87.93
Kanayama et al 99	Probabilistic model (ME + HPSG)	EDR (192,778)	88.55
Haruno et al 98	Probabilistic model (DT + Boosting)	EDR (50,000)	85.03
Fujio et al 98	Probabilistic model (ML)	EDR (190,000)	86.67

Table 4: Comparison with the related work

Kanayama et al. (2000) use an HPSG-based Japanese grammar to restrict the candidate dependencies. Their model uses at most three candidates restricted by the grammar as features; the nearest, the second nearest, and the farthest from the modifier. Thus, their model can take longer context into account, and disambiguate complex dependency relations. However, the features are still static, and dynamic features are not used in their model. We cannot directly compare their model with ours because they use a different corpus, EDR corpus, which is ten times as large as the corpus we used. Nevertheless, they reported an accuracy 88.55%, which is worse than our model.

Haruno et al. (99) report that using the EDR Corpus for their training and testing, they achieve around 85.03% accuracy with Decision Tree and Boosting. Although Decision Tree can take combinations of features as SVMs, it easily overfits on its own. To avoid overfitting, Decision Tree is usually used as an weak learner for Boosting. Combining Boosting technique with Decision Tree, the performance may be improved. However, Haruno et al. (99) report that the performance with Decision Tree falls down when they added lexical entries with lower frequencies as features even using Boosting. We think that Decision Tree requires a careful feature selection for achieving higher accuracy.

6 Conclusion

We presented a new Japanese dependency parser using a cascaded chunking model which achieves 90.46% accuracy using the Kyoto University Corpus. Our model parses a sentence deterministically only deciding whether the current segment modifies the segment on its immediate right hand side. Our model outperforms the previous probabilistic model with respect to accuracy and efficiency. In addition, we showed that dynamic features significantly contribute to improve the performance.

References

Steven Abney. 1991. Parsing By Chunking. In *Principle-Based Parsing*. Kluwer Academic Publishers.

Masakazu Fujio and Yuji Matsumoto. 1998. Japanese Dependency Structure Analysis based on Lexicalized Statistics. In *Proceedings of EMNLP '98*, pages 87–96.

Msahiko Haruno, Satoshi Shirai, and Yoshifumi Ooyama. 1999. Using Decision Trees to Construct a Practical Parser. *Machine Learning*, 34:131–149.

Hiroshi Kanayama, Kentaro Torisawa, Yutaka Mitsuishi, and Jun'ichi Tsujii. 2000. A Hybrid Japanese Parser with Hand-crafted Grammar and Statistics. In *Proceedings of the COLING 2000*, pages 411–417.

Taku Kudo and Yuji Matsumoto. 2000. Japanese Dependency Structure Analysis based on Support Vector Machines. In *Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 18–25.

Sadao Kurohashi and Makoto Nagao. 1997. Kyoto University text corpus project. In *Proceedings of the ANLP, Japan*, pages 115–118.

Kiyotaka Uchimoto, Satoshi Sekine, and Hitoshi Isahara. 1999. Japanese Dependency Structure Analysis Based on Maximum Entropy Models. In *Proceedings of the EACL*, pages 196–203.

Kiyotaka Uchimoto, Masaki Murata, Satoshi Sekine, and Hitoshi Isahara. 2000. Dependency model using posterior context. In *Procedings of Sixth International Workshop on Parsing Technologies.*

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. Wiley-Interscience.