A Unified Single Scan Algorithm for Japanese Base Phrase Chunking and 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 describe an algorithm for Japanese analysis that does both base phrase chunking and dependency parsing simultaneously in linear-time with a single scan of a sentence. In this paper, we show a pseudo code of the algorithm and evaluate its performance empirically on the Kyoto University Corpus. Experimental results show that the proposed algorithm with the voted perceptron yields reasonably good accuracy.

1 Introduction

Single scan algorithms of parsing are important for interactive applications of NLP. For instance, such algorithms would be more suitable for robots accepting speech inputs or chatbots handling natural language inputs which should respond quickly in some situations even when human inputs are not clearly ended.

Japanese sentence analysis typically consists of three major steps, namely morphological analysis, *bunsetsu* (base phrase) chunking, and dependency parsing. In this paper, we describe a novel algorithm that combines the last two steps into a single scan process. The algorithm, which is an extension of Sassano's (2004), allows us to chunk morphemes into base phrases and decide dependency relations of the phrases in a strict left-to-right manner. We show a pseudo code of the algorithm and evaluate its performance empirically with the voted perceptron on the Kyoto University Corpus (Kurohashi and Nagao, 1998).

2 Japanese Sentence Structure

In Japanese NLP, it is often assumed that the structure of a sentence is given by dependency relations among *bunsetsu*s. A *bunsetsu* is a base phrasal unit and consists of one or more content words followed by zero or more function words.

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

Figure 1: Sample sentence (bunsetsu-based)

In addition, most of algorithms of Japanese dependency parsing, e.g., (Sekine et al., 2000; Sassano, 2004), assume the three constraints below. (1) Each bunsetsu has only one head except the rightmost one. (2) Dependency links between bunsetsus go from left to right. (3) Dependency links do not cross one another. In other words, dependencies are projective.

A sample sentence in Japanese is shown in Figure 1. We can see all the constraints are satisfied.

3 Previous Work

As far as we know, there is no dependency parser that does simultaneously both bunsetsu chunking and dependency parsing and, in addition, does them with a single scan. Most of the modern dependency parsers for Japanese require *bunsetsu* chunking (base phrase chunking) before dependency parsing (Sekine et al., 2000; Kudo and Matsumoto, 2002; Sassano, 2004). Although word-based parsers are proposed in (Mori et al., 2000; Mori, 2002), they do not build bunsetsus and are not compatible with other Japanese dependency parsers. Multilingual parsers of participants in the CoNLL 2006 shared task (Buchholz and Marsi, 2006) can handle Japanese sentences. But they are basically word-based.

Proceedings of the ACL-IJCNLP 2009 Conference Short Papers, pages 49–52,
Suntec, Singapore, 4 August 2009.

	Meg	ga	kare	ni	ano	pen	wo	age-ta.
	Meg	subj	him	to	that	pen	acc	give-past.
ID	0	1	2	3	4	5	6	7
Head	1	7	3	7	6	6	7	-
Type	B	D	B	D	D	B	D	-

Figure 2: Sample sentence (morpheme-based). "Type" represents the type of dependency relation.

4 Algorithm

4.1 Dependency Representation

In our proposed algorithm, we use a morpheme-based dependency structure instead of a bunsetsu-based one. The morpheme-based representation is carefully designed to convey the same information on dependency structure of a sentence without the loss from the bunsetsu-based one. The rightmost morpheme of the bunsetsu t should modify the rightmost morpheme of the bunsetsu u when the bunsetsu t modifies the bunsetsu u. Every morpheme except the rightmost one in a bunsetsu should modify its following one. The sample sentence in Figure 1 is converted to the sentence with our proposed morpheme-based representation in Figure 2.

Take for instance, the head of the 0-th bunsetsu "Meg-ga" is the 4-th bunsetsu "age-ta." in Figure 1. This dependency relation is represented by that the head of the morpheme "ga" is "age-ta." in Figure 2.

The morpheme-based representation above cannot explicitly state the boundaries of bunsetsus. Thus we add the type to every dependency relation. A bunsetsu boundary is represented by the type associated with every dependency relation. The type "D" represents that this relation is a dependency of two bunsetsus, while the type "B" represents a sequence of morphemes inside of a given bunsetsu. In addition, the type "O", which represents that two morphemes do not have a dependency relation, is used in implementations of our algorithm with a trainable classifier. Following this encoding scheme of the type of dependency relations bunsetsu boundaries exist just after the morphemes that have the type "D". Inserting "|" after every morpheme with "D" of the sentence in Figure 2 results in Meg-ga | kare-ni | ano | pen-wo | age-ta. This is identical to the sentence with the bunsetsu-based representation in Figure 1.

Input: w_i: morphemes in a given sentence.
N: the number of morphemes.
Output: h_j: the head IDs of morphemes w_j.
t_j: the type of dependency relation. A possible value is either "B", "D", or "O".
Functions: Push(i, s): pushes i on the stack s.
Pop(s): pops a value off the stack s.
Dep(j, i, w, t): returns true when w_j should modify w_i. Otherwise returns false. Sets always t_j.

```
procedure Analyze(w, N, h, t)
var s: a stack for IDs of modifier morphemes
begin
  Push(−1, s);   { −1 for end-of-sentence }
  Push(0, s);
  for i ← 1 to N − 1 do begin
    j ← Pop(s);
    while (j ≠ −1
      and (Dep(j, i, w, t) or (i = N − 1)) ) do
    begin
      h_j ← i; j ← Pop(s)
    end
    Push(j, s); Push(i, s)
  end
end
```

Figure 3: Pseudo code for base phrase chunking and dependency parsing.

4.2 Pseudo Code for the Proposed Algorithm

The algorithm that we propose is based on (Sassano, 2004), which is considered to be a simple form of shift-reduce parsing. The pseudo code of our algorithm is presented in Figure 3. Important variables here are h_j and t_j where j is an index of morphemes. The variable h_j holds the head ID and the variable t_j has the type of dependency relation. For example, the head and the dependency relation type of "Meg" in Figure 2 are represented as $h_0 = 1$ and $t_0 =$ "B" respectively. The flow of the algorithm, which has the same structure as Sassano's (2004), is controlled with a stack that holds IDs for modifier morphemes. Decision of the relation between two morphemes is made in *Dep()*, which uses a machine learning-based classifier that supports multiclass prediction.

The presented algorithm runs in a left-to-right manner and its upper bound of the time complexity is $O(n)$. Due to space limitation, we do not discuss its complexity here. See (Sassano, 2004)

for further details.

5 Experiments and Discussion

5.1 Experimental Set-up

Corpus For evaluation, we used the Kyoto University Corpus Version 2 (Kurohashi and Nagao, 1998). The split for training/test/development is the same as in other papers, e.g., (Uchimoto et al., 1999).

Selection of a Classifier and its Setting We implemented a parser with the voted perceptron (VP) (Freund and Schapire, 1999). We used a polynomial kernel and set its degree to 3 because cubic kernels proved to be effective empirically for Japanese parsing (Kudo and Matsumoto, 2002). The number of epoch T of VP was selected using the development test set. For multiclass prediction, we used the pairwise method (Kreßel, 1999).

Features We have designed rather simple features based on the common feature set (Uchimoto et al., 1999; Kudo and Matsumoto, 2002; Sassano, 2004) for bunsetsu-based parsers. We use the following features for each morpheme:

1. major POS, minor POS, conjugation type, conjugation form, surface form (lexicalized form)
2. Content word or function word
3. Punctuation (periods and commas)
4. Open parentheses and close parentheses
5. Location (at the beginning or end of the sentence)

Gap features between two morphemes are also used since they have proven to be very useful and contribute to the accuracy (Uchimoto et al., 1999; Kudo and Matsumoto, 2002). They are represented as a binary feature and include distance (1, 2, 3, 4 – 10, or 11 $\leq$), particles, parentheses, and punctuation.

In our proposed algorithm basically two morphemes are examined to estimate their dependency relation. Context information about the current morphemes to be estimated would be very useful and we can incorporate such information into our model. We assume that we have the j-th morpheme and the i-th one in Figure 3. We also use the $j-n, ..., j-1, j+1, ..., j+n$ morphemes and the $i-n, ..., i-1, i+1, ..., i+n$ ones, where n is the size of the context window. We examined 0, 1, 2 and 3 for n.

Measure	Accuracy (%)
Dependency Acc.	93.96
Dep. Type Acc.	99.49
Both	93.92

Table 1: Performance on the test set. This result is achieved by the following parameters: The size of context window is 2 and epoch T is 4.

	Bunsetsu-based	Morpheme-based
Previous	88.48	95.09
Ours	NA	93.96

Table 2: Dependency accuracy. The system with the previous method employs the algorithm (Sassano, 2004) with the voted perceptron.

5.2 Results and Discussion

Accuracy Performances of our parser on the test set is shown in Table 1. The dependency accuracy is the percentage of the morphemes that have a correct head. The dependency type accuracy is the percentage of the morphemes that have a correct dependency type, i.e., "B" or "D". The bottom line of Table 1 shows the percentage of the morphemes that have both a correct head and a correct dependency type. In all these measures we excluded the last morpheme in a sentence, which does not have a head and its associated dependency type.

The accuracy of dependency type in Table 1 is interpreted to be accuracy of base phrase (bunsetsu) chunking. Very accurate chunking is achieved.

Next we examine the dependency accuracy. In order to recognize how accurate it is, we compared the performance of our parser with that of the parser that uses one of previous methods. We implemented a parser that employs the algorithm of (Sassano, 2004) with the commonly used features and runs with VP instead of SVM, which Sassano (2004) originally used. His parser, which cannot do bunsetsu chunking, accepts only a chunked sentence and then produces a bunsetsu-based dependency structure. Thus we cannot directly compare results with ours. To enable us to compare them we gave bunsetsu chunked sentences by our parser to the parser of (Sassano, 2004) instead of giving directly the correct chunked sentences

Window Size	Dep. Acc.	Dep. Type Acc.
0 ($T = 1$)	82.71	99.29
1 ($T = 2$)	93.57	99.49
2 ($T = 4$)	93.96	99.49
3 ($T = 3$)	93.79	99.42

Table 3: Performance change depending on the context window size

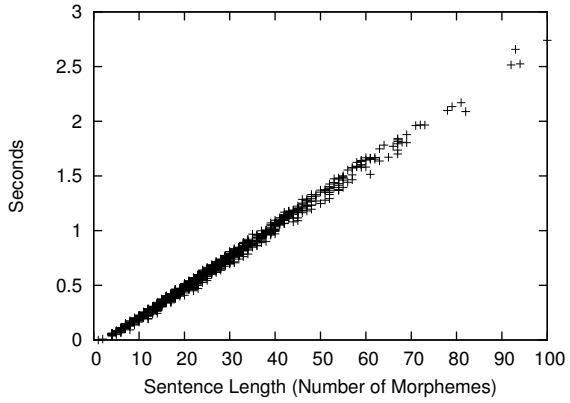

Figure 4: Running time on the test set. We used a PC (Intel Xeon 2.33 GHz with 8GB memory on FreeBSD 6.3).

in the Kyoto University Corpus. And then we received results from the parser of (Sassano, 2004), which are bunsetsu-based dependency structures, and converted them to morpheme-based structures that follow the scheme we propose in this paper. Finally we have got results that have the compatible format and show a comparison with them in Table 2.

Although the bunsetsu-based parser outperformed slightly our morpheme-based parser in this experiment, it is still notable that our method yields comparable performance with even a single scan of a sentence for dependency parsing in addition to bunsetsu chunking. According to the results in Table 2, we suppose that performance of our parser roughly corresponds to about 86–87% in terms of bunsetsu-based accuracy.

Context Window Size Performance change depending on the size of context window is shown in Table 3. Among them the best size is 2. In this case, we use ten morphemes to determine whether or not given two morphemes have a dependency relation. That is, to decide the relation of morphemes j and i $(j < i)$, we use morphemes $j-2, j-1, j, j+1, j+2$ and $i-2, i-1, i, i+1, i+2$.

Running Time and Asymptotic Time Complexity We have observed that the running time is proportional to the sentence length (Figure 4). The theoretical time complexity of the proposed algorithm is confirmed with this observation.

6 Conclusion and Future Work

We have described a novel algorithm that combines Japanese base phrase chunking and dependency parsing into a single scan process. The proposed algorithm runs in linear-time with a single scan of a sentence.

In future work we plan to combine morphological analysis or word segmentation into our proposed algorithm. We also expect that structure analysis of compound nouns can be incorporated by extending the dependency relation types. Furthermore, we believe it would be interesting to discuss linguistically and psycholinguistically the differences between Japanese and other European languages such as English. We would like to know what differences lead to easiness of analyzing a Japanese sentence.

References

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of CoNLL 2006*, pages 149–164.

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

U. Kreßel. 1999. Pairwise classification and support vector machines. In B. Schölkopf, C. J. Burges, and A. J. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*, pages 255–268. MIT Press.

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

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

S. Mori, M. Nishimura, N. Itoh, S. Ogino, and H. Watanabe. 2000. A stochastic parser based on a structural word prediction model. In *Proc. of COLING 2000*, pages 558–564.

S. Mori. 2002. A stochastic parser based on an SLM with arboreal context trees. In *Proc. of COLING 2002*.

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

S. Sekine, K. Uchimoto, and H. Isahara. 2000. Backward beam search algorithm for dependency analysis of Japanese. In *Proc. of COLING-00*, pages 754–760.

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