Corpus-based Dependency Analysis of Japanese Sentences using Verb Bunsetsu Transitivity

Daisuke Kawahara and **Sadao Kurohashi**

Graduate School of Informatics, Kyoto University
Yoshida-Honmachi, Sakyo-ku, Kyoto, 606–8501, Japan
{kawahara,kuro}@pine.kuee.kyoto-u.ac.jp

Abstract

In a Japanese sentence, the structural ambiguity becomes very serious when it contains many verb *bunsetsu*s. This paper proposes a parsing method based on possibilities of verb *bunsetsu* dependencies which are learned from a syntactically annotated corpus. By assuming the transitivity of verb *bunsetsu* dependencies, this method can avoid the data sparseness problem. We incorporate this corpus-based method with a rule-based parser KNP (Kurohashi and Nagao, 1994), and show its effectiveness.

1 Japanese Grammar and Task Definition

Let us first introduce Japanese grammar briefly. The structure of a Japanese sentence can be described well by the dependency relation between *bunsetsu*s. A *bunsetsu* is a basic unit in Japanese language, consisting of one or more content words and the following zero or more function words. The English equivalent for a *bunsetsu* would be a small noun phrase, a prepositional phrase, and a verb phrase consisting of auxiliary verbs and a main verb, and so on. Japanese language is head-final, that is, a *bunsetsu* depends on another *bunsetsu* to its right (not necessarily the adjacent *bunsetsu*).

There are two classes of *bunsetsu*, *noun bunsetsu* (NB) and *verb bunsetsu* (VB), depending on whether its content word is a noun or a verb. The function word or the ending form of a *bunsetsu* shows whether it should modify (depend on) NB or VB. Consequently, there are four classes of a *bunsetsu* as follows:

VB-modifying VB
eg. *kake*(stem of 'write') - *ba*(ending which indicates if-clause),
kaku(stem of 'write') - *to*(postposition which indicates when-clause)

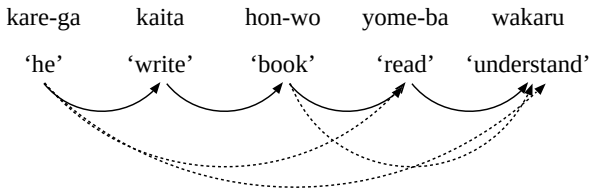
NB-modifying VB
eg. *kai*(stem of 'write') - *ta*(ending which indicates embedded sentence)

VB-modifying NB
eg. *kare*('he') -*ga*(nominative case marker),
hon('book') -*wo*(accusative case marker)

NB-modifying NB
eg. *kare*('he') -*no*('of')

The structural constraint on Japanese sentences is only the above distinction of four *bunsetsu* classes, which is too weak to detect a unique structure of a sentence, and often causes a lot of possible structures.

For example, the following example sentence has many possible dependency structures (solid arrows show correct dependencies; dotted arrows show other possibilities):



(Translation: If you read the book written by him, You can understand.)

A very powerful preference rule for this problem is that every bunsetsu depends on the nearest possible head. For example, all the dependency relations in the above example meet the rule. This rule, however, does not hold very well for VB-modifying VBs. A VB-modifying

VB often depends on much further *bunsetsu* in a sentence. That is, the analysis of VB-modifying VBs' heads is the key in Japanese sentence parsing. This paper proposes a corpus-based method for handling this problem.

Let us define some terminologies, here. We call generalized *bunsetsu* in which only function words or ending forms are remained, a *bunsetsu type*. For example, a *bunsetsu* type of "*yome-ba*" is $<\cdots ba>$. A set of several *bunsetsu* types are called a *bunsetsu class*.

2 Corpus-based Analysis of Verb *Bunsetsu* Dependencies

Minami proposed a preference rule about VB dependencies (Minami, 1993). He classified several types into three classes depending on their strength, e.g., $<\cdots keredo>$ 'although $\cdots$' is classified into the strongest class, $<\cdots tsutsu>$ 'while $\cdots$' is classified into the weakest class. Based on his VB classification, he claimed that a weaker VB cannot contain a stronger VB in its scope, that is, a stronger VB does not depend on a weaker VB.

Minami's claim seems true. However, his three-level VB classification was too coarse to handle a strength order among VBs precisely. Accordingly, KNP and Shirai et al. introduced more detailed classes and their strength order based on human intuition (Shirai et al., 1995). However, the more precise the rule system is, the more complicated it is, and the more expensive its maintenance is. This is an intrinsic problem in rule-based methods.

If a syntactically annotated corpus is available, it is not necessary to introduce VB classes, nor to rely on human intuition; the strength order among VB types can be learned directly. That is, if we find a VB type depends on another VB type in a syntactically annotated corpus, we can say that the former is weaker than the latter.

This paper proposes a parsing method based on VB types' dependencies which are learned from a syntactically annotated corpus. This method has the following two advantages:

- Because it doesn't classify VBs and doesn't determine the strength order manually, it doesn't have arbitrariness.
- Corpus-based methods intrinsically have the data sparseness problem. Assuming the transitivity of VB type dependencies, this problem can be avoided.

Table 1: Examples of VB types

examples	VB types
kaki-nagara	$<\cdots nagara>$
kaki-tsutsu	$<\cdots tsutsu>$
kaita-nara	$<\cdots nara>$
kaita NB	*<NB-modifying VB>*
kake-ba	$<\cdots ba>$
kaita-keredo	$<\cdots keredo>$
kaita-node	$<\cdots node>$
kaita-to	*<...to (quotation)>*
kaita.	*<EOS*[a]*>*

[a]EOS means a *bunsetsu* of the end of a sentence.

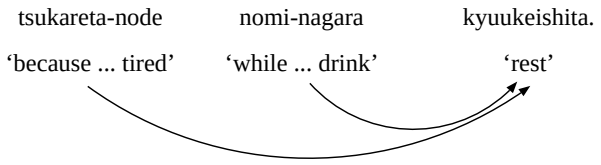
Although Minami didn't mention explicitly, some stronger VB can work as a *barrier* of dependency. Suppose a sentence like *hon-wo* 'book' *yomi-nagara*(A) 'read' *gakkou-ni* 'school' *itta-to,*(B) 'go' *hanashita.* 'say'. VB type $<\cdots$'*nagara*'$>$ (A) can depend on VB type $<\cdots$'*to*' $>$ (B), but type A cannot depend on the further VB than B actually. This means that VB type B is a barrier to VB type A. Such a information is an important cue to reduce the ambiguity of dependencies.

2.1 Learning of VB Type Dependencies

First, VB types are labeled depending on function words or the ending forms. VB types are distinguished by whether a VB is followed by a comma or not. There are 277 VB types in total. Table 1 shows examples of VB types.

The dependencies learned from a corpus are between two VB types (Figure 1).

Then, we count dependencies between two VB types in a sentence. When a VB type (VB_{type1}) goes over a latter VB type (VB_{type2}), depending on a further VB, we also count *going over* relation between the VB_{type1} and the VB_{type2}.



For example, from the above sentence, the following dependencies are learned.

dependent	head	depend/go over
<···node>	<EOS>	1/0
<···node>	<···nagara>	0/1
<···nagara>	<EOS>	1/0

2.2 Dependency Analysis Algorithm

After the learning process in the previous section, two VB types (the former one, VB_{typeA}, and the latter one, VB_{typeB}) can have one of the following four dependency patterns:

depend:found and go-over:found
VB_{typeA} can both depend on and go over VB_{typeB}. In the sense of Minami's claim, VB_{typeA} is weaker than VB_{typeB}.

depend:found and go-over:not-found
In this case, we interpret that VB_{typeB} is a barrier to VB_{typeA}. That is, when a sentence is something like "... VB_{typeA} ... VB_{typeB} ...", we consider that VB_{typeA} depends on VB_{typeB} or closer possible head.

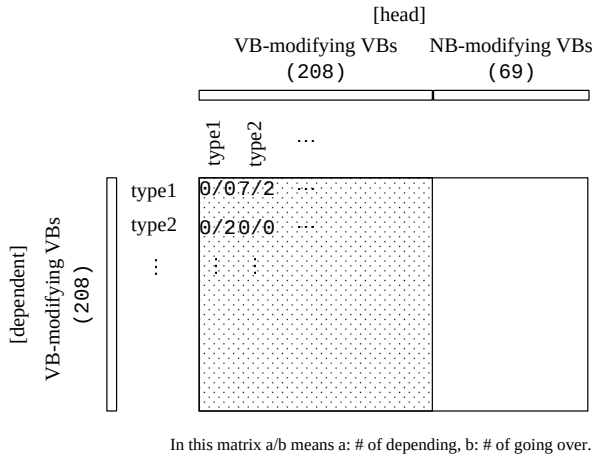


Figure 1: Relations between dependents and heads

depend:not-found and go-over:found
In this case, we interpret that VB_{typeA} is stronger than VB_{typeB} in the sense of Minami's claim, and VB_{typeA} cannot depend on VB_{typeB}.

depend:not-found and go-over:not-found
In this case, because of lack of data, dependency possibility between VB_{typeA} and VB_{typeB} cannot be estimated. We handle this case to be the same as depend:not-found and go-over:found, that is, VB_{typeA} is stronger than VB_{typeB}.

By using these criteria, we can reduce the ambiguity of VB-modifying VBs' dependencies.

The parsing system is based on the grammars of KNP, which is a dependency structure analyzer based on dependency grammar formalism. And the information learned from a training corpus is used to decide a head of a VB-modifying VB. If there are many head candidates, the nearest one is selected as a head.

2.3 Filling up and Cleaning up of VB Type Dependencies

There are two problems in the simple learning method described in Section 2.1:

- Noise (an exceptional dependency which holds in a special context, or an annotated error)
- Data sparseness

To solve these problems, we introduce two procedures below.

Filling up
In general, corpus-based methods have the data sparseness problem. To cope with this problem, we fill up VB type dependencies which doesn't exist in the training corpus by assuming the transitivity. That is, if there is a dependency between VB type X and VB type Y, and VB type Y and VB type Z, then we guess that VB type X can depend on VB type Z. In other words, we assume that we found a dependency between VB type X and VB type Z in the corpus.

Cleaning up

Noise in the corpus, an exceptional dependency or an annotated error, might cause side effects in parsing. Hence, we remove dependencies which make side effects by the following procedure.

First, the training corpus is parsed using the simply learned data. Counting the correct and erroneous numbers of each VB type dependency, if the ratio of the erroneous number against the correct number is bigger than some threshold, this dependency should be deleted from the learned data. A preliminary experiment shows that it is the most appropriate for the threshold to be set 1:1 (i.e., the erroneous number is equal to the correct number).

3 Experiments

3.1 Methods of Experiments

We used the Kyoto University Corpus (Kurohashi and Nagao, 1998) for learning. This corpus consists of about 30,000 sentences of newspaper articles. It contains manually cleaned part-of-speech tags and dependency tags.

First, the corpus is divided into two segments, the training and the test corpus, in several ways like cross-validation. Then, VB types' dependencies are learned from the training corpus, the testing corpus is parsed by the algorithm in Section 4.2, and the parse results are compared with the manually tagged structures.

When calculating a parsing accuracy, we forget the last and the second last *bunsetsu*, since the last one does not depend on any *bunsetsu* (the head of a sentence) and the second last must depend on the last one. Then a parsing accuracy is calculated as follows:

$$accuracy = \frac{\#\text{ of } bunsetsu\text{s whose head is correct}}{\#\text{ of } bunsetsu\text{s}}$$

3.2 Experimental Result

Table 2 shows the total result. This method's parsing accuracy is a little bit better than KNP. In this method, we achieve the same accuracy, as we use KNP, without the preference rules between VBs.

Table 3 shows the number of learned dependencies and parsing accuracies of one test set, when the filling up and the cleaning up of VB type dependencies are repeated. There are 208 dependent VB types and 277 head VB types, accordingly there are 57616 kinds of possible VB type dependencies. In the simple learning, 2212 dependencies were learned, and their accuracy was 80.6%. Finally, 17552 dependencies were learned, and their accuracy became 85.3%. This table shows that repeating the filling up and the cleaning up of VB type dependencies was very effective.

Table 2: Accuracies of dependencies - (learning twice)

	All	VB-modifying VBs
KNP	21764/24146 (0.901)	3534/4254 (0.831)
our method	21772/24146 (0.902)	3551/4254 (0.835)

Table 3: The number of learned dependencies and VB accuracy

	number	VB accuracy
simple learning	2212	1078/1337 (0.806)
Cleaning up 1	1953	1127/1337 (0.843)
Filling up 1	8585	-
Cleaning up 2	8186	1140/1337 (0.853)
Filling up 2	16923	-
Cleaning up 3	16391	1140/1337 (0.853)
Filling up 3	18095	-
Cleaning up 4	17552	1140/1337 (0.853)

4 Related Work

There have been many research activities in corpus-based Japanese parsing, which handle VB dependencies as one of dependency relations and do not pay special attention to VB dependencies (Fujio and Matsumoto, 1998; Shirai et al., 1998).

Nishiokayama proposed a statistical method of dependency analysis of VBs (Nishiokayama et al., 1998). His method concentrated on VB dependencies, and constructed decision lists automatically using detailed features concerning VBs.

These conventional methods, however, did not take account of Minami's claim, that there is a strength order among VBs, and a stronger VB does not depend on a weaker VB. On the

other hand, we exploit this characteristic, that is, the VB transitivity, which enables us to avoid the data sparseness problem.

5 Conclusion

This paper proposed a parsing method based on possibilities of VB type dependencies which are learned from an annotated corpus given syntactic structures. In this method, we obtained the same accuracy, as we use KNP, without the preference rules between VBs.

References

Masakazu Fujio and Yuji Matsumoto. 1998. Japanese dependency structure analysis based on lexicalized statistics. In *Proceedings of the 3rd Conference on Empirical Methods in Natural Language Processing*, pages 88–96.

S. Kurohashi and M. Nagao. 1994. A syntactic analysis method of long japanese sentences based on the detection of conjunctive structures. *Computational Linguistics*, 20(4).

S. Kurohashi and M. Nagao. 1998. Building a japanese parsed corpus while improving the parsing system. In *Proceedings of The First International Conference on Language Resources & Evaluation*, pages 719–724.

Fujio Minami. 1993. *Outline of contemporary Japanese grammar*. Taisyukan Syoten.

Shigeyuki Nishiokayama, Takehito Utsuro, and Yuji Matsumoto. 1998. Extracting preference of dependency between japanese subordinate clauses from corpus. In *Technical Report of the Institute of Electoronics, Information and Communication Engineers NLC98-11*, pages 31–38.

Satoshi Shirai, Satoru Ikehara, Akio Yokoo, and Junko Kimura. 1995. A new dependency analysis method based on semantically embedded sentence structures and its performance on japanese subordinate clauses. *Transactions of Information Processing Society of Japan*, 36(10):2353–2361.

Kiyoaki Shirai, Kentaro Inui, Takenobu Tokunaga, and Hozumi Tanaka. 1998. An empirical evaluation on statistical parsing of japanese sentences using lexical association statistics. In *Proceedings of the 3rd Conference on Empirical Methods in Natural Language Processing*, pages 80–87.