# KN Parser : Japanese Dependency/Case Structure Analyzer

Sadao Kurohashi and Makoto Nagao

Dept. of Electrical Engineering, Kyoto University
Yoshida-honmachi, Sakyo, Kyoto, 606, Japan
kuro@kuee.kyoto-u.ac.jp

## Abstract

This paper presents the KN parser, a Japanese dependency/case structure analyzer. It performs well in parsing Japanese sentences by taking account of certain characteristics in Japanese language. Several unique methods used in the parser are discussed and preliminary evaluation results on hundreds of Japanese sentences are given.

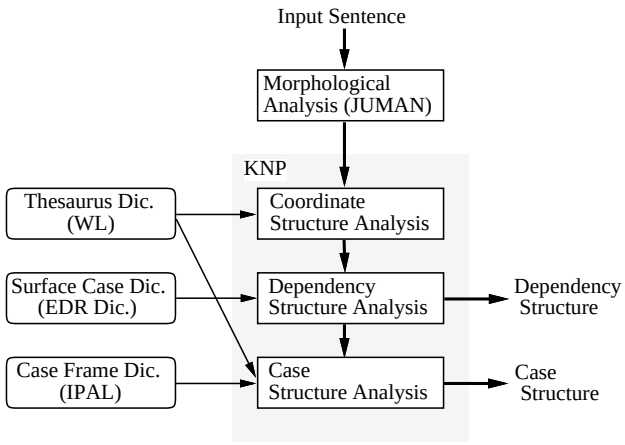



Figure 1: Overview of KN parser.

## 1 Introduction

This paper presents the KN Parser (Kurohashi-Nagao Parser), a dependency grammar parser that takes advantage of several unique methods that have been recently developed [1]. Figure 1 shows the overall flow of the system when parsing a sentence. One of the unique and most effective parts of the parser is its first component, namely, the coordinate structure analysis component. This component drastically reduces structural ambiguity in a sentence containing coordinate structures, making the subsequent analysis much easier.

Basically the KN parser works as a dependency structure analyzer based on dependency grammar formalism. Using a wide-coverage surface case dictionary and heuristic rules on syntactic information, the parser is able to handle large, real world texts. As a case structure analyzer the KN parser remains a prototype system at present because of the small number of entries in the case frame dictionary. However, since its example-based case structure analysis method has been shown to work well in preliminary experiments, we are planning to make it more practical by enlarging the case frame dictionary.

## 2 Characteristics of Japanese Language

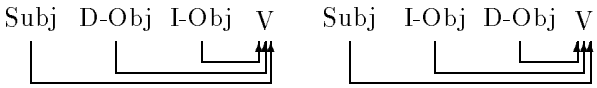
The Japanese language is quite different from English and many other European languages. Certain characteristics such as free word order, ellipsis and the use of coordinate structures must be considered when building a parser to work on real world texts [2].

### 2.1 Free Word Order and Varieties of Ellipsis

The Japanese language allows more freedom in the word order of sentences, as well as allowing the omission of various components in a sentence. While phrase structure grammars (PSG) have been popular in the analysis of English, they have difficulty in handling these phenomena. Because PSGs presuppose the word (phrase) order as specified in the grammar rules, they do not work well in languages which have free word order. Furthermore, to use a PSG in the analysis of a language which allows a variety of ellipses, we would have to prepare not only the rules which have ev-

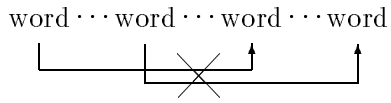
ery component in a rule, but also a number of rules which are missing various components. Because it is quite difficult to specify in what condition a certain element can be omitted, the PSG would be almost impossible to build, and the concept of grammatical restriction would not hold.

Considering these problems, we have adopted the dependency grammar (DG) formalism as a basis for our parser. DGs clarify which word modifies or depends on which, and does not say anything about subject, object, etc.. This property is useful for word-order change in Japanese because the modifier-modifiee relation is not influenced by the word-order as shown in the following.




In addition, DGs are not affected by omitted words; the analysis process is always the same.

A serious problem for DGs is handling ambiguity problems, in which a word modifies two or more words which follow in a sentence. This problem is relieved to a certain extent by the so-called non-crossing condition in the Japanese language, which prohibits any two modifier-modifiee relations from crossing with each other as shown in the following diagram.




The remaining ambiguity is solved by heuristic rules and/or by information in a case frame dictionary in the KN parser (see Sec. 4).

### 2.2 Coordinate Structures

Almost all current parsing systems fail in the analysis of long sentences, i.e., a sentence composed of more than seventy Japanese characters. Long sentences, particularly in Japanese, very often contain coordinate structures (CSs): coordinate noun phrases and/or coordinate predicative clauses. The latter are called "Renyoh chuushiho". They modify nouns in embedded sentences, and are also used to connect two or more sentences. This form is very often used in Japanese and is a major cause of structural ambiguity. Therefore an important step in the analysis of a long sentence is to identify such CSs. Serious attempts have been made to solve this problem by writing many grammar rules to check something like semantic similarities which may exist in the head nouns or main verbs of CS, but there have been no significant improvements so far.

What we have developed in our parser instead is an algorithm based on the assumption that parallel phrases/clauses/sentences will have a certain similarity in the sequence of words and their grammatical structures as a whole [3]. By finding the two most similar word-strings, we can successfully detect the scopes of CS, based on this assumptions. Because we had to compare two word-strings of arbitrary lengths in a sentence, we adopted a dynamic programming method to calculate overall similarity values for all possible two word-strings of arbitrary lengths and then selected the best one. This algorithm was inserted inbetween the morphological analysis and the dependency analysis in our parsing system, and is very useful in the analysis of real world texts.

## 3 Resources used in the KN Parser

One characteristic of the KN parser is that it uses almost all the current publicly available resources: morphological analyzer, thesaurus dictionary, surface case dictionary, and case frame dictionary. These resources are described briefly below.

**Morphological analyzer — JUMAN** [4]
Japanese raw sentences do not have words separators such as spaces. The processing of a Japanese sentence has to start with a morphological analysis that includes word segmentation. We have also developed a publicly available morphological analyzer, JUMAN. The KN parser receives as input the morph-string of a sentence output by JUMAN.

**Thesaurus dictionary — Word List by Semantic Principles (WL)** [5]
Similarity calculation is one of the most important components in the KN parser. We calculate similarity values between two word-strings in the coordinate structure analysis, and similarity values between example sentences in a case frame dictionary and an input sentence in the case structure analysis. These similarities are basically composed of semantic similarity values between two words which can be calculated by using the thesaurus dictionary, WL, constructed by the National Language Research Institute. WL has a six

Table 1: Examples of case frames for "HAIRU" in IPAL.

| ⟨*Case markers*⟩ | ⟨*Sem. markers*⟩ | ⟨*Examples*⟩ | ⟨*Deep cases*⟩ |
|---|---|---|---|
| **Case frame 1** (*Meaning* : Enter) | | | |
| N1-GA | [HUM/ORG/ ANI/PRO] | {KARE(*he*), IKKOU(*party*), NORA-NEKO(*cat*), FUNE(*ship*)} | agent |
| N2-KARA (optional) | [LOC] | {MADO(*window*), URAGUCHI(*rear-gate*)} | locational source |
| N3-NI/E | [LOC] | {KYOUSHITSU(*classroom*), DAIDOKORO (*kitchen*), *MINATO(port)*} | locational goal/directional |
| **Case frame 2** (*Meaning* : Be added to food or drink) | | | |
| N1-NI | [PRO] | {COFFEE(*coffee*), CAKE(*cake*)} | non-locational goal |
| N2-GA | [CON] | {SATOU(*sugar*), MILK(*milk*), CHEESE(*cheese*), DOKU(*poison*)} | object |
| **Case frame 3** (*Meaning* : Be reflected) | | | |
| N1-NI | [PRO/ABS] | {SAKUHIN(*work*), HOUKOKU-SHO (*report*), *AN(proposal)*} | non-locational locative |
| N2-GA | [MEN] | {KANGAE(*thought*), IKEN(*opinion*), DOKUDAN(*arbitrariness*)} | object |

layer abstraction hierarchy and covers more than 30,000 words. We can consider the most specific common layer between two words in the thesaurus tree as a indication of their semantic similarity.

**Surface case dictionary — EDR Japanese Word Dictionary** [6]
During the dependency structure analysis, the parser consults surface case information in the EDR Japanese Word Dictionary to eliminate inappropriate connections between predicates and case components in a sentence. Since the dictionary has rich vocabulary (200,000 words), it is very useful for widening the coverage of the parser.

**Case frame dictionary — IPA Lexicon of the Japanese Language for computers (IPAL)** [7]
During the case structure analysis, the parser consults the case frame dictionary, IPAL, constructed by Information-technology Promotion Agency, Japan. In IPAL, there are 861 basic verbs entries, and each entry has various sub-entries according to differences in its meaning and syntax. Case frame information is given for each sub-entry, consisting of the meaning of a verb, its case markers (postpositions), semantic markers, examples, and correspondences to deep cases for each case slot (see Table 1).

# 4 System Description

As is shown in Figure 1, the KN parser consists of three steps: coordinate structure analysis, dependency structure analysis, and case structure analysis. The outlines of each step are given in the following sections (the details are given in papers [1, 3, 8]).

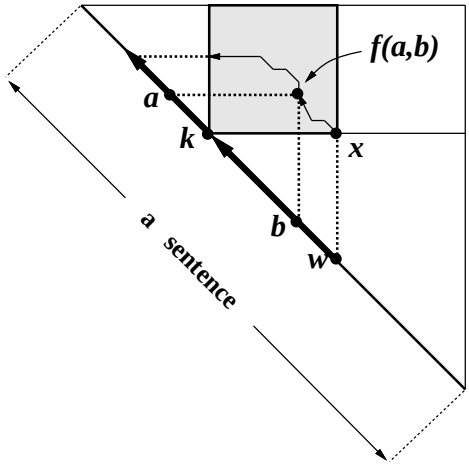



Figure 2: Dynamic programming method to recognize a coordinate structure in a sentence.

## 4.1 Coordinate Structure Analysis

The first step of the KN parser is the coordinate structure (CS) analysis. The main idea of the analysis is to determine similarities between two arbitrary word strings by using a dynamic programming method. This method is shown conceptually in Figure 2. An example is given in Figure 3-A. We place a sentence on a diagonal edge of a triangle indexed by word unit (or, in the case of Japanese, by **bunsetsu** unit which is composed of a content word and suffix words). Then we determine which units contain suffix words or

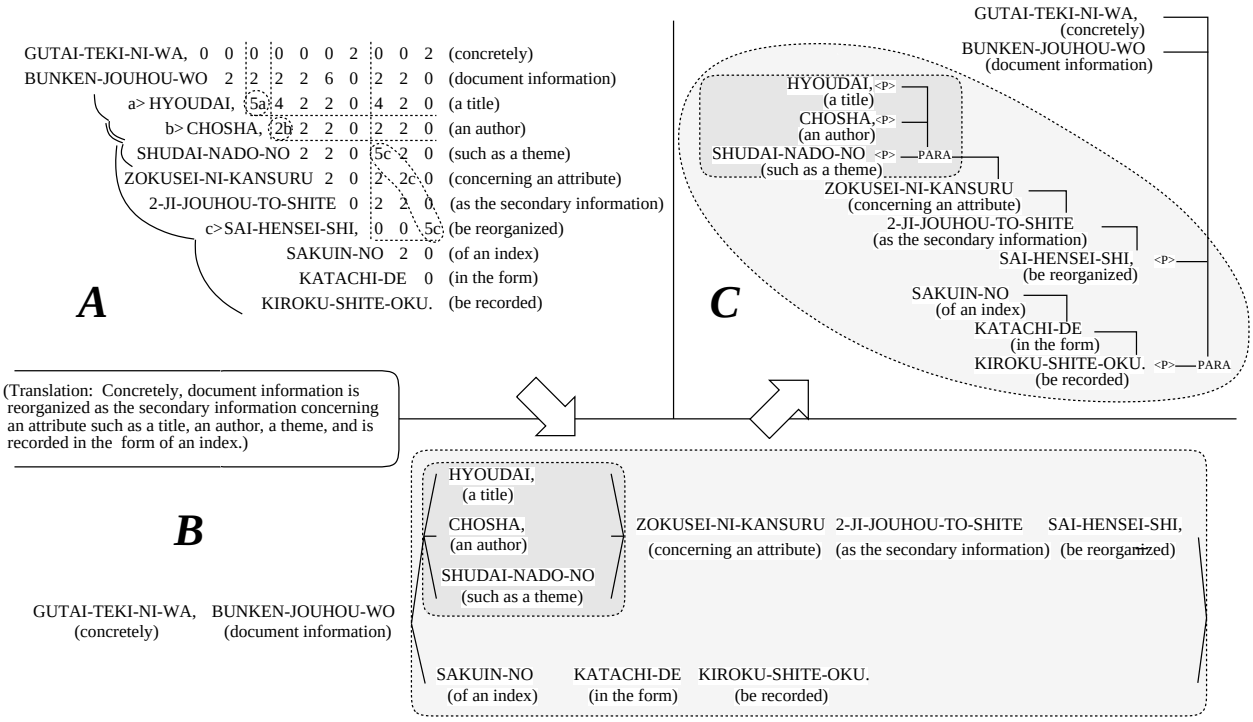



Figure 3: An example of analyzing a long sentence.

special symbols which indicate the possible existence of CSs on both sides of them. "TO(and)", "MATAWA(or)", "," etc. are typical words and symbols, and hereafter are called **keys for CSs**. The keys in Figure 3-A are indicated by >, for example **a**>, **b**>, and **c**>. In each crossing position of two words in the triangle as shown in Figure 2, the similarity value of the two words is determined according to part-of-speech matching, exact matching, and closeness in the thesaurus tree (in Figure 2, for two words $a$ and $b$, their similarity value $f(a,b)$ is given at the crossing point). Then we begin the similarity calculation of two word strings of arbitrary length, starting one from a key ($k$ in Figure 2) and going to the left, and starting the other from an arbitrary word ($w$ in Figure 2) which is to the right of the key and going to the left. A dynamic programming method is used to obtain the best pass inside the shaded square in Figure 2 starting from the point $x$ and going to the upper left. The best pass is obtained in the following manner. Starting at point $x$ the similarity values are added together whenever a step path is available diagonally to the upper left, but no addition is done if the step path is horizontal or vertical. This process is done for all the possible paths from $x$ until the pass reaches a grid point on the vertical line through $k$. The maximum addition value and its corresponding path are then recorded at position $x$. This process is performed for all the words ($w$) to the right of $k$. The word $w$ whose $x$ has the largest value among all the possible words $w$ is regarded as the headword of the second word string of the CS. The leftmost word of the first word string of the CS is the one which corresponds to the upper-left end of the best path from $x$. In Figure 3-A the best chosen paths for the keys **a**>, **b**>, and **c**> are marked in the triangle.

As a result of parsing these CSs in a sentence, we can get a reduced sentence form (Figure 3-B). This form has less structural ambiguity than the original form of the sentence, and makes the following processes — dependency structure analysis and case structure analysis — much easier.

## 4.2 Dependency Structure Analysis

The next step in the parser is the analysis of the dependency structure. First, each conjunct of the CSs is analyzed. Because the pre- and post-conjuncts have their own consistent structures, they are parsed independently into dependency trees. If there are two or more CSs in a nested structure in a sentence, each CS is analyzed from the innermost CS in the order of nesting levels. Finally, the main sentential components are analyzed. For example, in the sentence in Fig-

ure 3 the inner CS is analyzed first, resulting in a new node with conjoined nouns assigned to it (the darkly hatched part in Figure 3-C: 'PARA' is the new node). Next, the pre- and post-conjuncts of the outer CS are analyzed and are transformed into dependency trees, and another new node is created (the lightly hatched part in Figure 3-C). Finally, the whole sentence is analyzed, and its dependency tree is completed.

The remaining problem in the dependency structure analysis stage is how to handle structural ambiguities within a certain range of bunsetsu-sequence in a reduced sentence. The KN parser detects the unique dependency structure of a sentence by using the following rules:

**Non-crossing condition**
This condition prohibits any two modifier-modifiee relations from crossing each other.

**Constraint under surface case information**
By consulting the surface case information in the EDR Japanese Word Dictionary, inappropriate modifier-modifiee relations between a predicate and case components are eliminated.

**Nearest modifiee principle**
When a structural ambiguity remains through the above two rules, that is, a bunsetsu still can depend on two or more bunsetsus, it is basically treated as depending on the nearest possible modifiee since this is very common in a Japanese sentence. However, if a bunsetsu which has two or more possible modifiees is one of the following types:

- a predicate of a conditional or causal form,
- a bunsetsu accompanying a comma, or
- a bunsetsu containing a topic-marking postposition "WA",

it is treated as depending on a modifiee-candidate further away (see [1] for the details).

When applied to the restricted range of bunsetsu-sequence, this set of rules works very well and the dependency structure of a sentence can be obtained with a high accuracy.

Since the EDR Japanese Word Dictionary has rich vocabulary (200,000 words) and other rules depend only on syntactic information, the KN parser as a dependency structure analyzer has wide coverage, and hence it is useful as a basic process in doing advanced NLP, like extracting information by analyzing large corpora.

### 4.3 Case Structure Analysis

From the standpoint of machine translation or other natural language processing, it is necessary to clarify the role of each word or phrase in a sentence. Case structure representation of a sentence is surely one of the best forms that meet that requirement. In the KN parser, the case structure analysis can follow the dependency structure analysis.

The case structure analysis is performed by consulting a case frame dictionary, IPAL. One difficulty in the case structure analysis is that since a verb often has two or more meanings (usages), it is necessary to select a proper case frame for a verb in an input sentence from among the case frames available in the dictionary. A conventional method for solving this problem is the use of *selectional restrictions*, where the category of the nouns which are able to fill in the case slot is specified by semantic markers, such as *human*, *animate*, *action*, and so on. However, through experiments on IPAL, we have determined that semantic markers in IPAL (19 different semantic markers) are too coarse to select a proper case frame for an input sentence, and that an example-based method, in which the case frame whose example is most similar to the input sentence is selected, yields much better results. Accordingly, the KN parser selects a proper case frame based not on semantic markers but on examples assigned to case slots in IPAL. In brief, the similarity value between an input sentence and an example sentence in a case frame is calculated as the sum of similarity values between case components in the input and examples of the matching case slots by the equality of case markers in the case frame.

For example, let us suppose that we select a proper case frame for the following input sentence out of the case frames in Table 1:

| KISHA-GA | TON'NERU-NI | HAIRU. |
|---|---|---|
| (*train*) | (*tunnel*) | (*enter*) |
| [PRO] | [LOC/PRO] | |

If we were to use semantic markers, case frame 3 would be removed by comparing the semantic marker PRO(product) of "KISHA(*train*)" with the semantic marker MEN(mental) of the case slot "N2-GA", but the inappropriate case frame 2 would be selected together with the proper case

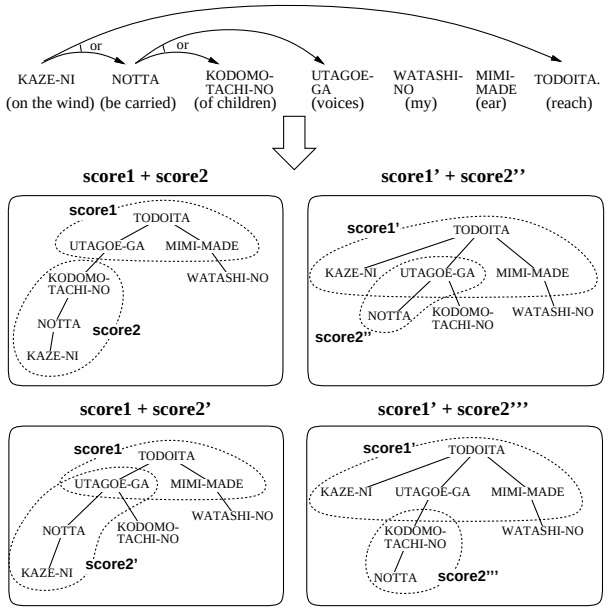



Figure 4: Ranking all possible readings by the sum of the scores for their case structures.

frame 1. With the example-based method, however, the input sentence is much more similar to examples of case frame 1 than to examples of case frame 2, the proper case frame 1 would be selected easily by matching the input with the examples in these case frames.

Example sentences in the case frame dictionary can be used not only for case frame selection but also for structural disambiguation. Structural disambiguation can be done by considering the best matching score (similarity value), which was used in selecting a proper case frame for a verb and its case components, as the score which shows the degree of *appropriateness* for the case structure. When there are two or more readings (dependency structures) for a sentence because of structural ambiguity, the best reading (defined as the correct or the most plausible reading) is the one in which all verbs in the sentence govern appropriate case components, and their case structures have high scores. This means that the best reading of a sentence can be selected by evaluating the sum of the scores for the case structures of all verbs in the sentence (Figure 4).

Since this method of structural disambiguation is more reliable than the use of a simple heuristic rule (the nearest modifiee principle in Sec. 4.2), whenever the case structure analysis is performed, the dependency structure analysis component does not use the nearest modifiee principle, and instead produces all of the possible readings. The case structure analysis component selects proper case frames for all the argument structures in all readings, and finally selects the reading which has the maximum addition score for its case structures. Even if a sentence is long, since the boundaries of CSs in the sentence are already detected, the possible readings have already been limited enough that the parser can evaluate all of them.

Table 2: Coordinate and dependency structure analysis result on bunsetsu level.

| | # of bunsetsus | # of correct analyses | success ratio |
|---|---|---|---|
| KBs | 215 | 190 | 88% |
| Others | 1736 | 1685 | 97% |

The current case frame dictionary, IPAL, contains less than 1,000 verbs, so the case structure analysis is not very useful at present. However, users can provide case frame information in the same framework as IPAL to make the case structure analysis more practical for their corpora.

# 5 Experimental Evaluation

We summarize the preliminary experiments on hundreds of Japanese sentences reported so far [1].[1]

## 5.1 Coordinate Structure and Dependency Structure Analysis

As an experimental evaluation of coordinate structure analysis and dependency structure analysis, we report the results of analyzing 150 test sentences. The test sentences are longer and more complex than sentences in common usage. The test set contains 50 sentences composed of 30 to 50 characters, 50 sentences of 50 to 80 characters, and 50 sentences of over 80 characters. They were collected at random from scientific/technical articles.

First, we divided all the bunsetsus in the 150 sentences into two groups: bunsetsu containing keys of CSs (KB), and other bunsetsus. Then, we manually checked both the KB set to determine whether the scopes of their corresponding

[1] For these experiments we intentionally chose difficult test sentences to test the limits of our parser. In parsing ordinary sentences, the success ratio should be much higher.

Table 3: Coordinate and dependency structure analysis result on sentence level.

| | **a** | **b** | **c** | **a** | **b** | **c** | **a** | **b** | **c** | **a** | **b** | **c** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| # of characters in a sentence | 30–50 | | | 50–80 | | | 80–149 | | | Total | | |
| Sentences with no CS | 29 | – | 25 | 10 | – | 5 | 4 | – | 3 | 43 | – | 33 |
| Sentences with CSs | 21 | 15 | 14 | 40 | 34 | 30 | 46 | 37 | 20 | 107 | 86 | 64 |
| Total | 50 | – | 39 | 50 | – | 35 | 50 | – | 23 | 150 | – | 97 |

**a** : The number of sentences which were classified into this category.
**b** : The number of sentences in which all the CSs were detected correctly.
**c** : The number of sentences whose whole dependency structures were analyzed correctly.

CSs were analyzed correctly, and the other bunsetsus to determine whether their modifiees were analyzed correctly. The results are shown in Table 2. This table shows that our dynamic programming method has a high success ratio in finding CSs, and that the simple heuristic rules for modifier-modifiee relations are good enough to analyze each phrase/clause of the CSs internally and the sentence in which CSs have already been merged into nodes.

We then classified the 150 sentences by their length and according to whether or not they contain CSs. We manually checked whether CSs in each sentence were detected correctly, if they exist, and whether their dependency structures were analyzed correctly (Table 3). Here, the analysis is regarded as a success only when all the CSs in a sentence were detected correctly, and all the modifier-modifiee relations in a sentence were detected correctly. Considering that the test sentences have considerable length and there are many candidate modifiees for each bunsetsu in a sentence, the total success ratio at the sentence level, 65% (97/150), can be considered fairly good. Although one third of the dependency structures after this analysis process included some errors, their major structures were detected correctly in most cases. This can be seen from the high success ratio in the bunsetsu-level evaluation in Table 2.

## 5.2 Case Structure Analysis

We report two experiments to illustrate the effectiveness of the example-based case structure analysis in the KN parser.

In the first experiment we selected proper case frames for simple sentences. First, we chose 20 verb entries from IPAL, all of which have many case frames (the average number of case frames for a verb is 15.8). We asked a language-trained person to compose a set of about 450 simple test sentences each of which includes one of these verbs. Particular care was taken to ensure that the test sentences were distributed over all the different usages of these verbs by consulting large Japanese dictionaries. Then we analyzed the test sentences with the KN parser and checked by hand whether the proper case frames were selected. The results are as follows:

1) There is only one best match case frame, and this is a proper case frame to be selected. 62.7%
2) There are several best match case frames, one of which is a proper case frame to be selected. 12.0%
3) A proper case frame was not included in the best match case frames. The matching value was lower for a proper case frame. 23.2%
4) There is no proper case frame in IPAL dictionary which corresponds to an input sentence. 2.0%

When we analyzed the same test sentences with semantic markers in IPAL, the percentage of selecting a proper case frame uniquely (case 1 above) was only 29.9%. The main reason for incorrect analysis in the example-based method was insufficiency of examples in IPAL. Some case slots have only one or two examples. This problem, however, can be solved simply by adding the wrongly analyzed sentences as new examples of their proper case frames.

The second experiment was done to determine the effectiveness of the example-based structural disambiguation in the KN parser. We again had a language-trained person compose a set of about 250 complex test sentences (by using IPAL verb entries) each of which includes one or more clausal modifiers and has structural ambiguities. Then we analyzed these test sentences with our method and evaluated the analysis results from the viewpoint of structural disambiguation (Table 4). The

Table 4: Structural disambiguation result in the case structure analysis.

| | # of sentences | # of correct analyses | success ratio |
|---|---|---|---|
| **A** | 196 | 183 | 93% |
| **B** | 37 | 33 | 89% |
| Total | 233 | 216 | 93% |

**A** : Sentences in whose best reading the nearest modifiee principle holds.

**B** : Sentences in whose best reading the nearest modifiee principle does not hold.

success ratio of obtaining a correct dependency structure by this method is very high. If we just use the nearest modifiee principle instead, all the sentences in row 'A' in Table 4 can be analyzed correctly, but those in row 'B' cannot, so the total success ratio is just 84% (196/233). Accordingly, the example-based structural disambiguation is approximately 9% accurate than the method by the nearest modifiee principle.

# 6 Computer Platform

The KN parser was developed on the Sun SPARC station series, written in C. It is a free software, has no limitation, and is available through anonymous ftp from `pine.kuee.kyoto-u.ac.jp` (`130.54.31.90`).
The file name is `/pub/knp/knp.tar.Z` archived by unix `tar` and `compress`. It requires `gcc` and `gmake` to compile the system. The complete system uses 16MB of space, including the compiled database for dictionaries used in the KN parser. Some brief documentation is included in the system archive file, and papers describing details of the system are available through ftp from the same site (`/pub/knp/knppaper.tar.Z`).

The KN parser uses the resources described in Section 3, and they should be gotten first. Our system includes tools to integrate those dictionaries into the KN parser (using unix `ndbm`).

Ftp instructions and more information can be obtained by mailing requests to `knp@pine.kuee.kyoto-u.ac.jp`.

# 7 Conclusion

We discussed the several components of the KN parser and gave preliminary evaluation results. More work is required to improve our parser. As a dependency structure analyzer, it is important to evaluate and improve the parser through experiments on a large amounts of texts. Investigating the analysis result of large texts will also suggest other interesting targets for real natural language understanding. As a case structure analyzer, we need to compile case frame information on a large number of verbs to make the parser practical. We hope to enlarge the current case frame dictionary by increasing its entries semi-automatically.

# References


[1] S. Kurohashi, *Global Strategies for High Precision Analysis of Japanese Sentences*, PhD thesis, Kyoto University, 1993.

[2] M. Nagao, "Varieties of heuristics in sentence processing," In *Current Issues in Natural Language Processing: In Honour of Don Walker*, Giardini with Kluwer, 1994.

[3] S. Kurohashi and M. Nagao, "Dynamic Programming Method for Analyzing Conjunctive Structures in Japanese," In *Proceedings of 14th COLING*, Nantes, Vol.1, pp.170–176, 1992.

[4] S. Kurohashi, T. Nakamura, Y. Matsumoto and M. Nagao, "Improvements of Japanese morphological analyzer JUMAN," In *Proceedings of Int. Workshop on Sharable Natural Language Resources*, Nara, Japan, 1994.

[5] The National Language Research Institute, *Word List by Semantic Principles*, Shuuei Publishing, 1964.

[6] EDR, *EDR Electronic Dictionary Technical Guide*, TR-042, 1993.

[7] M. Hasimoto, W. Kuwahata, K. Murata, F. Aoyama and T.Tonoike, "IPA Lexicon of the Japanese language for computers," In *Proceedings of Int. Workshop on Sharable Natural Language Resources*, Nara, Japan, 1994.

[8] S. Kurohashi and M. Nagao, "Structural Disambiguation in Japanese by Evaluating Case Structures based on Examples in Case Frame Dictionary," In *Proceedings of Int. Workshop on Parsing Tech.*, Tilburg/Durbuy, 1993.