An Interactive Japanese Parser for Machine Translation

Hiroshi Maruyama
maruyama@jpntscvm.bitnet
Hideo Watanabe
watanabe@jpntscvm.bitnet
Shiho Ogino

IBM Research, Tokyo Research Laboratory
5-19 Sanbancho, Chiyoda-ku,
Tokyo 102 Japan

Abstract

In this paper, we describe a working system for interactive Japanese syntactic analysis. A human user can intervene during parsing to help the system to produce a correct parse tree. Human interactions are limited to the very simple task of indicating the modifiee (governor) of a phrase, and thus a non-expert native speaker can use the system. The user is free to give any information in any order, or even to provide no information. The system is being used as the source language analyzer of a Japanese-to-English machine translation system currently under development.

1 Introduction

Despite the long history of research and development, perfect or nearly perfect analysis of a fairly wide range of natural language sentences is still beyond the state of the art. The users of the existing batch-style machine translation systems are obliged to post-edit the machine-translated text even if it contains errors because of an analysis failure.

We have developed an interactive Japanese syntactic analysis system, JAWB (Japanese Analysis WorkBench), for a Japanese-to-English machine translation system. It can produce very reliable syntactic structures with the help of a human user. User interactions are limited to the very simple task of specifying the modifiee (governor) of a phrase, and thus a non-expert native speaker can use the system. The number of user interactions is minimized by using *constraint propagation* (Waltz 1975) to eliminate inconsistent alternatives.

One feature of our system not found in previous attempts (Kay 1973, Melby 1980, Tomita 1986) is that the user is completely free to give the system any information in any order. He also has the alternative of providing no information. In this case, the system runs fully automatically, although the quality of output may be degraded.

In the next section, we describe the system structure. Then in Section 3 we discuss the interactive dependency analysis, and show a sample session. Section 4 gives the results of evaluation of the system.

2 System Structure

The system structure of JAWB is shown in Figure 1. Japanese syntax analysis is divided into two parts: morphological analysis and dependency analysis.

An input sentence is first segmented into a sequence of linguistic units called *bunsetsu*, which can be roughly translated in English as *phrases*. Each bunsetsu, hereafter called a phrase, consists

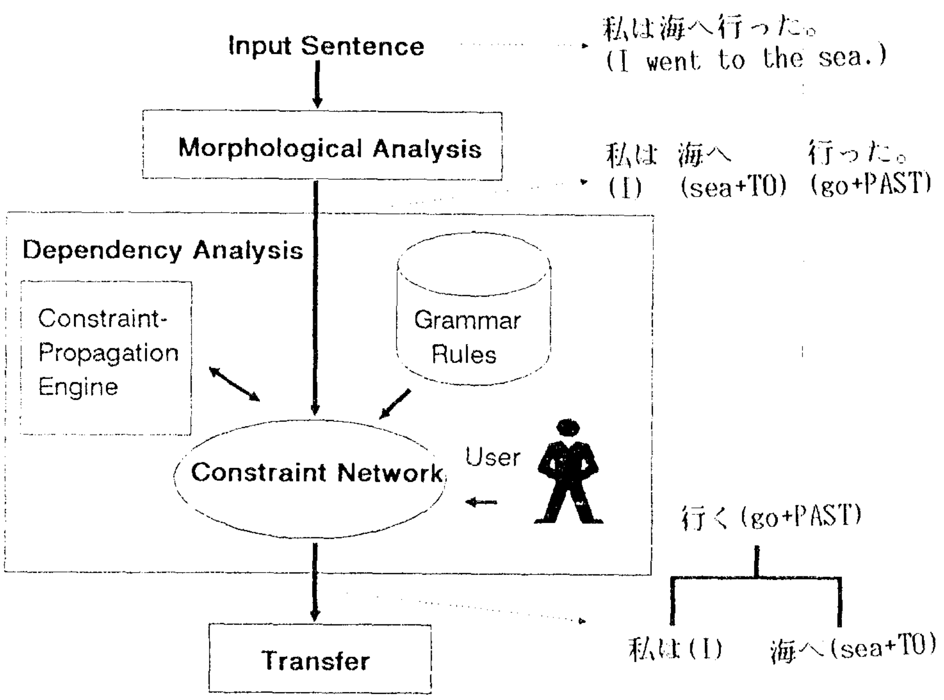


Figure 1: System structure

of one or more primitive words. The morphological analyzer analyzes a consecutive sequence of characters and identifies word and phrase boundaries. Japanese morphological analysis is a relatively well established technology (Maruyama et al. 1988) and intervention by the user is seldom required, although the system does provide a facility for this.

A Japanese syntactic structure is depicted by modifier-modifiee relationships between phrases. The dependency analyzer determines the modifiee of each phrase. This is the most difficult task and normally user interaction takes place at this stage. First, the system determines the modifiee candidates of each phrase by using the grammar rules, and builds a data structure called a *constraint network*. The grammar rules are based on Constraint Dependency Grammar (Maruyama 1990), and are essentially constraints between modifications. The constraint network holds the modifiee candidates of each phrase, and the grammatical constraints are posed between the candidates.

The system then proposes the most plausible reading and displays it on the screen along with the other possibilities. If the human user is satisfied with the proposal or does not want to make any decision, he tells the system to 'go ahead' and the proposal is passed through to the transfer component as the unique parsing result. Alternatively, the user can select an arbitrary phrase and choose its modifiee from the rest of the candidates. The system incorporates this information into the constraint network, makes another proposal, and shows it to the user. This process is iterated until no more ambiguity remains. During analysis, the *constraint propagation engine* keeps the constraint network locally consistent by using the *constraint propagation algorithm* (Waltz 1975).

Before the unique parse tree is submitted to the transfer component, JAWB performs some 'post processing' on the tree. This processing includes resolving remaining lexical ambiguities, giving grammatical relations such as SUBJ and DOBJ, and transforming a passive-voice structure into an active-voice structure. Since making such decisions requires expert knowledge about Japanese linguistics and/or the system's internal structure, it is preferable that this process is carried out automatically. Since correct modifier-modifiee relationships are given at the previous stage, this process makes few errors without human intervention.

```
$sent = {
  [phrase=1, string="あなたが"(anataga),
   cat=np, mcat=pred_modifier,
   modifiee={%2,3,4,5%},
   words= {
     [string="あなた"(anata),
      syn={%
        [pos=105,
         string="貴方"(you),
         sem=[sf={hum},caseframe={}]],
        [pos=105,
         string="彼方"(far off),
         sem=[sf={loc,con,abs},caseframe={}]]
     %}],
     [string="が"(ga),
      syn=[pos=75,string="が"(SUBJ)]]
  }],
  [phrase=2, string=" 昨日 "(kinou),
   cat=advp, mcat=pred_modifier,
   modifiee={%3,4,5%},

                    ⋮
```

Figure 2: Input to the dependency analyzer

3 Dependency Analysis

Let us consider sentence (1).

```
(1) あなたが   昨日       出会った
    Anataga   kinou      deatta
    you-SUBJ  yesterday  meet-PAST

    男を      見た。
    otokowo   mita.
    man-OBJ   see-PAST
```

Part of the input to the dependency analyzer for this sentence is shown in Figure 2. A sentence is a sequence of phrases, each of which is represented as a feature structure. Some of the values are enclosed by special brackets {% and %}, representing *disjunctions* or *choice points*. Phrase 1 in Figure 2, for example, contains two choice points, one for structural ambiguity (the *modifiee* slot) and the other for lexical ambiguity (the *syn* slot of the first word). In Japanese, every phrase except the last one modifies exactly one phrase on its right.[1] Therefore, the modifiee of phrase 1 is one of the four succeeding phrases.

The grammatical rules that we need here are as follows:

```
for X in $sent begin
 /* G1. pred_modifier modifies a pred */
 (X.mcat=pred_modifier =>
   $sent.(X.modifiee).cat in {vp,adjp,adjvp}
 ) &
 /* G2. noun_modifier modifies a noun */
 (X.mcat=noun_modifier =>
   $sent.(X.modifiee).cat in {np}
 )
end

for X,Y in $sent begin
 /* G3. modifications do not cross */
 X.phrase<Y.phrase & Y.phrase<X.modifiee =>
 Y.modifiee <= X.modifiee
end
```

According to the above rules, the modifiee (i.e., the governor) of phrase 1 (you-SUBJ) is either phrase 3 (meet-PAST) or phrase 5 (see-PAST), since phrase 1 is a predicate-modifier and phrases 3 and 5 are predicates. Similarly, phrase 2 can modify either phrase 3 or phrase 5. The values of the *modifiee* slot of each phrase thus become as follows:

```
phrase 1: modifiee={%3,5%}
phrase 2: modifiee={%3,5%}
phrase 3: modifiee={%4%}
phrase 4: modifiee={%5%}
```

Because modification links do not cross each other (by rule G3), the cases of phrase 1 modifying phrase 3 and phrase 2 modifying phrase 5 do not co-occur. Therefore, this sentence has three different readings, which correspond to (1-1) to (1-3):

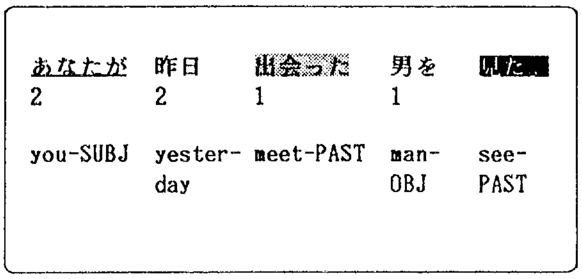
(1-1) (I) saw the man you met yesterday.

(1-2) You saw the man (I) met yesterday.

(1-3) Yesterday you saw the man (I) met.

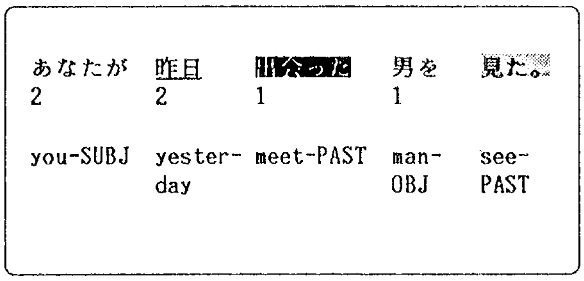
The system maintains these readings implicitly by having constraints between choice points. For example, the following *constraint ma-*

[1]This is a common view of Japanese syntax, although there are different views.



あなたが	昨日	出会った	男を	見た。
2	2	1	1	
you-SUBJ	yester-day	meet-PAST	man-OBJ	see-PAST

a. When the cursor is on phrase 1



あなたが	昨日	出会った	男を	見た。
2	2	1	1	
you-SUBJ	yester-day	meet-PAST	man-OBJ	see-PAST

b. When the cursor is on phrase 2

Figure 3:

trix is attached between the two choice points `$sent.1.modifiee` and `$sent.2.modifiee`:

`$sent.1.` `modifiee`	`$sent.2.` `modifiee`	value
3	3	1
3	5	0
5	3	1
5	5	1

By means of the constraint matrices, the system can defer the generation of individual parse trees until all structural ambiguities are resolved. The number of parse trees may combinatorially explode when the sentence becomes long. For example, sentences with more than 20 phrases are not rare and such sentences may have tens of thousands of parse trees.

User Interface

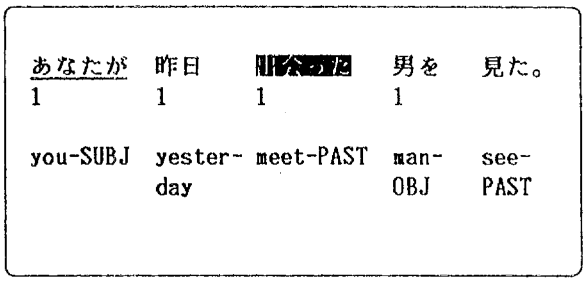
The essential portion of the user interface is shown in Figure 3. The system does not display the proposed modifiees of all the phrases at once. Instead, when the user moves the cursor to a phrase by using a mouse, the proposed modifiee and the other possible candidates are highlighted. In the figures, the current phrases pointed to by the cursor are underscored, the proposed modifiees are in reversed video, and the other modifiee candidates are in a shaded box. [2] The number appearing at the left lower corner of each phrase shows the number of modifiee candidates of the current phrase. If this number is one, the modifiee is uniquely determined. Otherwise, the modifiee of the phrase is ambiguous.

Figure 3-a shows the screen when the cursor is on phrase 1 (you-SUBJ). Phrase 1 can modify either phrase 3 or phrase 5, and the system's proposal is phrase 5. Figure 3-b shows the screen when the cursor is on phrase 2. By moving the cursor on the phrases, the user can check the current system proposal. If the user is satisfied with it, he indicates this by clicking a special 'go-ahead' icon. Otherwise, he has to select the proper candidates.

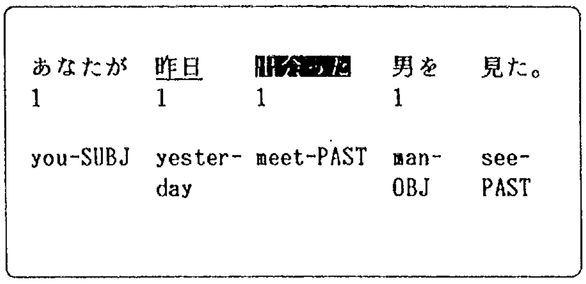
The user selects one of the ambiguous phrases by clicking the mouse, moves the cursor to its proper modifiee, and clicks the mouse again. The second click triggers the constraint propagation engine, and the updated situation is displayed instantaneously. Figure 4 shows the situation after the user has instructed the system that phrase 1 modifies phrase 3. The reader may notice that the modifiee of phrase 2 is also determined automatically because of constraint propagation.

During parsing, the user always has the initiative in the interaction. The user knows the exact sources of the structral ambiguity, and he can select any of them to give information to the system. This is in contrast to the previous systems, in which the user must answer system-generated queries one by one. The constraint propagation engine ensures that the given information is maximally used in order to minimize further interaction. The user also has the option of saying 'go-ahead'

[2]These are in different colors on the real screen.



あなたが	昨日	会った	男を	見た。
1	1	1	1	
you-SUBJ	yester-day	meet-PAST	man-OBJ	see-PAST



あなたが	昨日	会った	男を	見た。
1	1	1	1	
you-SUBJ	yester-day	meet-PAST	man-OBJ	see-PAST

Figure 4: Screens after specifying that phrase 1 modifies phrase 3

at any time, taking the default choices proposed by the system.

4 Evaluation

One of the claims of JAWB is that it can be used by non-expert users. To validate the claim, we conducted a comparative test with an expert user and a non-expert user. Figure 5 shows the results of the test. Subject A is one of the authors who actually developed the grammar. Subject B is a Japanese native speaker with no background in linguistics or computer science. Given an initial screen of dependency analysis, subject A spent 12.9 seconds on the average before making a correct parse tree. This period includes the time spent specifying the proper modifiees (1.1 times on average) and verifying the system proposals, but does not include overheads such as the time spent choosing a new sentence to be analyzed and waiting for the system to look up dictionaries from a disk. The same task took 18.8 seconds for subject B. The important point here is that although the performance is somewhat different, the parse trees generated by both subjects were essentially identical.[3] This means that, with a non-expert human user's help, JAWB is capable of producing very reliable parse trees fairly efficiently, although the efficiency can be increased by about 50% if an expert user uses it.

Another yardstick for evaluating the system is the accuracy of the initial proposals. From 1,089 test sentences taken from actual newspaper articles, JAWB generated correct initial proposals for 507 sentences (47%), which means that, if it is used in a full-automatic mode, its accuracy is 47%. On the other hand, the system rejected two sentences as ungrammatical, which means that for 99.8% of the test sentences, JAWB was capable of producing correct parse trees with appropriate user interaction.

[3]There were differences when the sentence was truly ambiguous, in which case even a human user could not resolve the ambiguity without the context knowledge.

5 Conclusion

JAWB is currently being used to accumulate correct parse trees for a corpus of texts. The accumulated data are vital for the development of our machine translation system for at least two reasons:

1. The transfer component, which generates an English syntactic structure from a Japanese syntactic structure, is difficult to develop without having enough error-free input data, that is, Japanese parse trees.

2. The accumulated parse trees are used as reliable linguistic data from which various statistical data are obtained in order to refine the grammar rules.

We believe that interactive source language analysis is a promising approach to practical machine translation not only because it may significantly reduce the task of post editing, which should be carried out by a professional translator, but also because the cost-saving effect is multiplied when the same text is translated into several different languages.

Sentence length	Ave. time (sec.)		Ave. # of interaction	
(# of phrases)	Subj. A	Subj. B	Subj. A	Subj. B
1 - 3	-	-	-	-
4 - 6	3.6	6.3	0.0	0.1
7 - 9	8.3	14.1	0.5	0.7
10 - 12	14.6	20.6	1.5	2.1
13 - 15	21.1	31.9	2.1	2.8
16 - 18	27.5	32.5	2.5	3.5
19 - 21	48.0	42.0	4.0	4.0
Ave. 9.8	12.9	18.8	1.1	1.5

Figure 5: User performance

Acknowledgements

The authors are grateful to Masayuki Morohashi, Hiroshi Kitamura, and Hiroshi Nomiyama for their valuable discussions and suggestions. The authors also would like to thank Michael McDonald for his help in preparing the manuscript.

References

1. Kay, Martin. 1973, "The MIND system," in Rustin, R. (ed.) *Natural Language Processing*, Algorithmics Press.
2. Maruyama, Hiroshi., 1990, "Structural disambiguation with constraint propagation," *Proc. of ACL Annual Meeting.*
3. Maruyama, Naoko; Morohashi, Masayuki; Umeda, Shigeki; Sumita, Eiichiro, 1988, "A Japanese sentence analyzer," *IBM Journal of Research and Development,* Vol. 32.
4. Melby, Alan. 1980, "ITS: Interactive translation system," *Proceedings of COLING '80.*
5. Tomita, Masaru. 1986, "Sentence disambiguation by asking," *Computers and Translation*, Vol. 1.
6. Waltz, David 1975, "Understanding line drawings of scenes with shadows," in: Winston, P.H. (ed.): *The Psychology of Computer Vision*, McGraw-Hill.