Automatic Extraction of Japanese Grammar from a Bracketed Corpus

Kiyoaki Shirai and **Takenobu Tokunaga** and **Hozumi Tanaka**
Department of Computer Science
Tokyo Institute of Technology
Tokyo, Japan
{kshirai,take,tanaka}@cs.titech.ac.jp

Abstract

In recent years, numerous attempts have been devoted to derive Context Free Grammars (CFGs) by using a large corpus. In this paper, we describe a method to extract a Probabilistic Context Free Grammar (PCFG) of Japanese from a bracketed corpus, and propose two methods to improve it. The experiments show that the extracted PCFG has a 94 % accept rate, 85 % brackets recall and 75 % brackets precision.

1 Introduction

There have been numerous attempts to derive a CFG by using a large corpus. The advantage of this approach includes: less human labor, broad coverage of the grammar and independence from individual introspection. In addition, it is easy to incorporate statistical features into the grammar. This may help to choose a correct parse tree.

Lari and Young (Lari and Young, 1990) suggested the use of the Inside-Outside algorithm for grammar inference. They assume the grammar is represented in the Chomsky normal form. Given a set of non-terminal and terminal symbols, they start with an initial grammar that is generated from every combination of symbols. Then they estimate the probabilities of the rules using the Inside-Outside algorithm, and remove the rules that have less probabilities than a certain threshold. Pereira and Schabes (Pereira and Schabes, 1992) modified the Inside-Outside algorithm for a partially bracketed corpus in order to reduce computing time. Kiyono and Tsujii (Kiyono and Tsujii, 1994a; Kiyono and Tsujii, 1994b) proposed a semi-automatic method to acquire new rules of an existing CFG. When a parse fails, they extract candidates of new rules from the results of partial parsing. The plausibility of the candidates is estimated from a training corpus, and only candidates that have higher plausibility are added to the grammar. Their method aims to extend the coverage of a given CFG with support of human decisions.

One of the drawbacks of these previous attempts is that their methods need much computing time. This would be an obstacle to deal with a large corpus. In addition, little attention has been paid to acquiring a Japanese grammar.

This paper proposes a method to extract a PCFG of Japanese language using a bracketed corpus with less computing time (Shirai et al., 1995). We describe the basic idea in section 2, then we propose two methods to improve the PCFG. One reduces the grammar size by removing redundant rules, as described in section 3; the other decreases the number of unnecessary parse trees, as described in section 4. In section 5, we show the results of experiments that evaluate the proposed methods.

2 Extracting PCFG

2.1 Training Corpus

We make use of 75,000 sentences in the EDR corpus (EDR, 1994), which are excerpted from newspaper articles and magazines. The sentences range from 5 words to 81 words in length. The average length is about 23 words. Each sentence is annotated with morphological and syntactical information, that is, each word is segmented, tagged with its part of speech (POS), and the syntactic structure is given by a skeletal tree. Figure 1 shows an example of a sentence. The meaning of this sentence is “The funding for the choral group

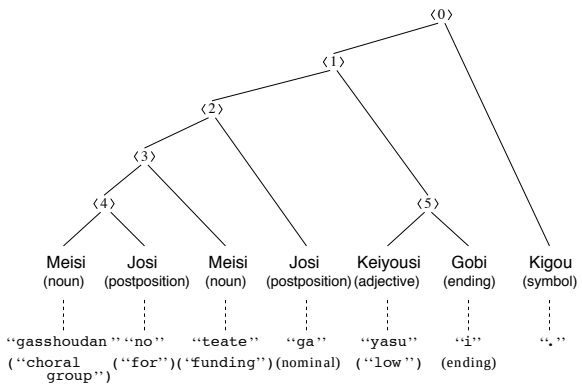

Figure 1: Example sentence in the EDR corpus

is low."

2.2 Extracting CFG rules

Each node other than any leaf in a skeletal tree can be transformed into a production rule that has the node itself on the left hand side (LHS) and the immediate children on the right hand side (RHS). Table 1 shows the rules that are extracted from the skeletal tree shown in Figure 1. Each bracketed number in the rules corresponds to an intermediate node in Figure 1. Note that the symbols in the RHS of these rules are lexical categories. In order to transform these rules into CFG rules, we need to assign appropriate non-terminal symbols to the bracketed numbers in Table 1. In other words, we can extract CFG rules by labeling intermediate nodes in skeletal trees.

Table 1: The rules derived from the skeletal tree

⟨5⟩	→	Keiyousi (adjective) Gobi (ending)
⟨4⟩	→	Meisi (noun) Josi (postposition)
⟨3⟩	→	⟨4⟩ Meisi (noun)
⟨2⟩	→	⟨3⟩ Josi (postposition)
⟨1⟩	→	⟨2⟩ ⟨5⟩
⟨0⟩	→	⟨1⟩ Kigou (symbol)

When labeling the intermediate nodes, we have to take account of the head [1] of a phrase. In contrast to English, Japanese is a head-final language. Thus, we can derive the LHS symbol from the right most symbol of the RHS. In general, we assign "Y-Ku" ("Y phrase") to ⟨i⟩ in the following rule:

$$\langle \mathrm{i} \rangle \rightarrow X_1\ X_2\ \cdots\ X_n\ Y$$

Here, ⟨i⟩ is an intermediate node, "X_i" and "Y" are lexical categories such as "Meisi" ("noun"), "Keiyousi" ("adjective") and "Josi" ("postposition"). Thus, we obtain the rule:

$$Y\text{-Ku} \rightarrow X_1\ X_2\ \cdots\ X_n\ Y.$$

Deriving "Y-Ku" from "Y" corresponds to raising the bar level. In fact, "Ku" means "phrase" in Japanese. For example, we assign ⟨4⟩ "Josi-Ku" because the right most symbol (head) of the second rule in Table 1 is "Josi." We have to assign the same symbol to ⟨4⟩ in the third rule of Table 1 at the same time, since it denotes the same node in Figure 1.

This labeling scheme is usually appropriate. However, there are some exceptions. In the EDR corpus, the POS "Kigou" ("symbol") is given to the symbols such as period, comma, question mark, parenthesis, etc. We can not regard "Kigou" as the head of phrases even if it is the right most POS of the rule. Another exception is the POS "Gobi" ("ending"). "Gobi" is assigned to the endings of verbs, auxiliary verbs and adjectives. It is not desirable to assign "Gobi-Ku" to a node whose right most child is "Gobi", because we can not distinguish these three categories. Therefore, when "Kigou" or "Gobi" appear at the right most position of a rule, we consider the previous POS as the head. For example, in the following rule, we regard "Keiyousi" ("adjective") as the head of the phrase and assign "Keiyousi-Ku" to ⟨i⟩.

⟨i⟩	→	Keiyousi (adjective)	Gobi (ending)
⇓			
Keiyousi-Ku	→	Keiyousi	Gobi

In order to extract CFG rules, we assign appropriate labels to intermediate nodes of the skeletal trees from bottom to up (i.e. from leaves to a root) according to the following procedure.

1. Find a node that has no unlabeled child. If there is no such node in the skeletal trees, go to 5.
2. Identify the head from amongst the children. This is done by scanning the children from right to left to find a symbol other than "Gobi" ("ending") and "Kigou" ("symbol").
3. Make a new label concatenating "-Ku" to the head, if the head symbol does not end with

[1] In this paper, the word "head" means a syntactic head, not a semantic head.

"-Ku." If the head symbol ends with "-Ku", the new label is equal to the head symbol, that is, it constitutes a right recursive rule.

4. Assign the designated new label to the node, except in the case of the root of the tree. Assign the root the label "S" (i.e. the start symbol). Go to 1.

5. Extract the CFG rules from the labeled trees.

One of the significant features of this method is that its time complexity is linear with respect to the total length of sentences. In addition, the procedure is a deterministic process. Table 2 shows the CFG rules extracted from the skeletal tree in Figure 1.

Table 2: The extracted CFG rules

Keiyousi-Ku	→	Keiyousi (adjective)	Gobi (ending)
Josi-Ku	→	Meisi (noun)	Josi (postposition)
Meisi-Ku	→	Josi-Ku	Meisi (noun)
Josi-Ku	→	Meisi-Ku	Josi (postposition)
Keiyousi-Ku	→	Josi-Ku	Keiyousi-Ku
S	→	Keiyousi-Ku	Kigou (symbol)

Next, we estimate the probabilities of the extracted rules in a very simple way. When we extract the rules from the bracketed corpus, we count the occurrence of the rule r, $C(r)$. For a rule $A \rightarrow \zeta_i$, its probability is estimated as follows.

$$P(A \rightarrow \zeta_i) = \frac{C(A \rightarrow \zeta_i)}{\sum_j C(A \rightarrow \zeta_j)}$$

3 Reducing the grammar size

In this section, we describe the method to reduce the size of the extracted grammar while retaining its coverage. First, we define a "redundant rule" as follows. The rule $A \rightarrow \zeta$ is redundant iff the non-terminal symbol A can be expanded into the symbol sequence ζ using the rules other than $A \rightarrow \zeta$. For example, in the following grammar, the rule r_a is redundant because non-terminal symbol A can be expanded into $BCDEF$ using the rules r_b, r_c, r_d. Thus, we can remove the rule r_a from the grammar without narrowing its coverage.

$$\begin{array}{lllllll} r_a: & A & \rightarrow & B & C & D & E & F \\ r_b: & A & \rightarrow & G & H & & & \\ r_c: & G & \rightarrow & B & C & & & \\ r_d: & H & \rightarrow & D & E & F & & \end{array}$$

The problem we must consider now is how to deal with the previously calculated figure for the occurrence of redundant rules. As we have mentioned above, we use the sum of rule occurrence to estimate the probability of the rule. Even though redundant rules may be removed, we have to take account of their occurrence. In the above example, the number of the occurrence of r_a should be added to that of r_b, r_c and r_d, because the structure subtended by r_a in the training corpus is replaced with that produced by the combination of r_b, r_c and r_d. Consider now the case where the following rules r'_b, r'_c, r'_d also exist in the grammar.

$$\begin{array}{lllll} r'_b: & A & \rightarrow & I & D & J \\ r'_c: & I & \rightarrow & B & C & \\ r'_d: & J & \rightarrow & E & F & \end{array}$$

Since A can also be expanded to $BCDEF$ by using r'_b, r'_c and r'_d, the occurrence figure for r_a must be distributed between $\{r_b, r_c, r_d\}$ and $\{r'_b, r'_c, r'_d\}$, in proportion to the number of the occurrence of r_b and r'_b.

The algorithm to remove redundant rules from a CFG is detailed below. In the algorithm, "R" denotes the original CFG, and "R_{new}" denotes the reduced grammar. The initial value of R_{new} is set empty.

1. Find the rule r_a in R, that has the longest RHS, and remove it from R.

2. Find all rule sets $\{r^j_b,\ r^j_{c1},\ \cdots,\ r^j_{cn}\}$ in R, which satisfy the following conditions:

$$\begin{array}{ll} r_a: & A \rightarrow \alpha^j_1\ \beta^j_1\ \alpha^j_2\ \cdots\ \alpha^j_n\ \beta^j_n\ \alpha^j_{n+1} \\ \hline r^j_b: & A \rightarrow \alpha^j_1\ B^j_1\ \alpha^j_2\ \cdots\ \alpha^j_n\ B^j_n\ \alpha^j_{n+1} \\ r^j_{c1}: & B^j_1 \rightarrow \beta^j_1 \\ & \vdots \\ r^j_{cn}: & B^j_n \rightarrow \beta^j_n \end{array}$$

   In the above, α^j_i, β^j_i are sequences of terminal and non-terminal symbols, and A, B^j_i are non-terminal symbols.

3. If no rule set is found in Step 2, r_a is not redundant and is added to R_{new}. Otherwise, r_a is a redundant rule, and we do not add it to R_{new}, but update the number of the

occurrence of the rules in the rule set $\{r_b^j, r_{c1}^j, \cdots, r_{cn}^j\}$ as follows.

for all i, j

$$C(r_b^j) \leftarrow C(r_b^j) + C(r_a) \times \frac{C(r_b^j)}{\sum_k C(r_b^k)}$$

$$C(r_{ci}^j) \leftarrow C(r_{ci}^j) + C(r_a) \times \frac{C(r_b^j)}{\sum_k C(r_b^k)}$$

4. If R is empty, then terminate, else go to 1.

4 Decreasing ambiguity

The extracted grammar described in the previous section gives a great number of parse trees for an input sentence. In this section, we will describe two methods of modifying the grammar in order to suppress unnecessary parse trees.

4.1 Structure of compound words

We will use the term "compound word" to refer to any constituent that consists of items of the same POS. In Figure 2, we show two examples of a "compound noun":

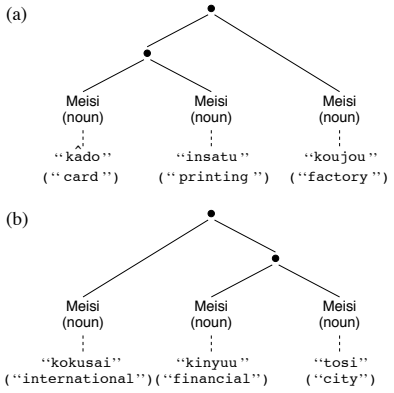

Figure 2: Examples of compound noun

They have the same sequence of POS, but have different structure. In general, it is difficult to determine the correct structure of compound nouns without regarding semantic information (Kobayashi et al., 1994). Therefore, parsing a sequence of "Meisi" ("noun") such as "Meisi Meisi Meisi" results in both structures (a) and (b). Increasing the length of the sequence would cause combinatorial explosion in the number of parse trees of distinct structure.

In order to suppress unnecessary parse trees, we make only a right linear binary branching tree (as shown in Figure 3) for a compound word in the syntactic analysis. We assume the correct structure is identified in the semantic analysis phase.

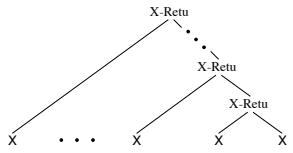

Figure 3: Right linear tree for compound word

We assign a new non-terminal symbol "*X*-Retu", where "Retu" means "sequence", to the root node and the intermediate nodes of a right linear tree whose leaves are all POS "*X*."

In order to obtain a right linear tree for a compound word, we make the following modification on the algorithm shown in section 2.2.

1. Find every subtree whose leaves are all the same POS X and replace it with the non-terminal symbol "*X*-Retu" in the training corpus.

2. Execute the algorithm in section 2.2.

3. For each subtree replaced in the Step 1, add to the grammar the following rule A with occurrence *1* and rule B with occurrence $n-2$, where n is the number of the leaves in the subtree.

   A : X-Retu $\rightarrow$ X X
   B : X-Retu $\rightarrow$ X X-Retu

A preliminary experiment with 10,000 sentences shows that this modification decreases the number of parse trees by 20 %.

4.2 Subcategorization of part of speech

The categorization of POSs in the EDR corpus is rather coarse. There are only 15 POSs. This is one of the reasons that the grammar gives so many parse trees. In this section, we propose to subcategorize some of the POSs into more fine-grained ones. In particular, we take two POSs, "Kigou" ("symbol") and "Josi" ("postposition").

4.2.1 Kigou (symbol)

As mentioned in section 2.2, various symbols such as period, comma, parentheses and so forth have POS "Kigou" ("symbol"). Among them,

period, question mark and comma appear frequently and play important roles in the syntactic structure. Thus, we subcategorize POS "Kigou" ("symbol") into the following three subcategories: "Bunmatu-Kigou" ("end-mark") for periods and question marks, "Touten" ("comma") for commas and "Kigou" ("symbol") for all remaining symbols.

4.2.2 Josi (postposition)

There are several kinds of "Josi" ("postposition") in Japanese, which generally indicate grammatical case, such as nominal, accusative and dative. In the EDR corpus, however, all the postpositions are given the same POS "Josi" ("postposition"). It is reasonable to distinguish them because they play very different roles in a sentence.

We assign a new POS "Josi-W" to postpositions, where "W" is the spelling of the postposition itself. For example, we assign POS "Josi-ga" to the postposition "ga" shown in Figure 1. This procedure can be done automatically.

The preliminary experiment with 10,000 sentences shows that subcategorizing these two POSs decreases the number of parse trees by 70 %. On the other hand, the number of the extracted rules grows about 5 times, but still remains a reasonable size.

5 Experiment

We conducted an experiment to evaluate our methods by using the EDR corpus described in section 2.1. Our experiment was executed on a Sun Sparc Station 10/51. Out of the 75,000 sentences in the EDR corpus, we used 67,500 for extracting PCFG and 7,500 as test sentences.

Table 3 shows the features of the extracted grammar. Removing the redundant rules decreases the grammar size by about 80 %.

Table 3: Extracted PCFG

the number of non terminal symbols	22
the number of terminal symbols	161
the number of rules (before removing redundant rules)	1781 (8011)

Next, we analyzed the test sentences by using a GLR parser (Tomita, 1986) with the extracted PCFG. In spite of our efforts to decrease the number of parse trees (section 4), the parser failed to analyze 20 % of the test sentences due to insufficient memory. Therefore, we introduced a pruning mechanism into the parser. At each reduce action, the probability of the subtree was calculated and those which had less probability than a certain threshold were discarded.

Table 4 shows the result of parsing test sentences [2]. The row "accept" shows the cases that the parser generates at least one parse tree. While, the row "reject" shows the cases that the parser generates no parse tree. The row "overflow" shows the cases that the parser aborts due to the insufficient memory. From the Table 4, we can see that the accept rate is nearly 94 %. The large part of the failure rate occurred due to insufficient memory, thus we consider that the extracted grammar has a broad coverage.

Table 4: The result of syntactic analysis

	without pruning	with pruning
accept	5897	6931
reject	12	15
overflow	1465	428

$$\text{accept rate} = \frac{6931}{7374} = 93.99\%$$

We evaluated the accuracy of parse trees on the basis of the number of compatible brackets that were consistent with the bracketings of the skeletal trees (Pereira and Schabes, 1992). First, we define "correct brackets" and "correct parse tree" as follows.

- Correct brackets is a pair of brackets that does not cross any bracket annotated in the test sentence.
- A correct parse tree is a parse tree that contains only correct brackets.

In order to evaluate the accuracy of the parse trees, we first chose the tree which had the fewest wrong brackets within the 15 most probable parse trees for each test sentence. Then, we counted the number of correct brackets and correct parse trees within these selected parse trees.

Table 5 shows the sentence accuracy, and the brackets recall and precision. Each of them are defined as follows:

[2] We could not parse all 7,500 test sentences due to the inconsistencies of the annotated tree.

sentence accuracy

$$= \frac{\text{No. of correct sentences}}{\text{No. of accepted sentences}}$$

brackets recall

$$= \frac{\text{No. of correct brackets}}{\text{No. of brackets in test sentences}}$$

brackets precision

$$= \frac{\text{No. of correct brackets}}{\text{No. of brackets in all parse trees}}$$

Table 5: The accuracy of parse trees

	without pruning	with pruning
Correct	1731	1731
Incorrect	4166	5200
Total	5897	6931

$$\text{sentence accuracy} = \frac{1731}{6931} = 24.97\%$$

$$\text{brackets recall} = \frac{105485}{124377} = 84.81\%$$

$$\text{brackets precision} = \frac{105485}{141067} = 74.78\%$$

The brackets recall (about 85 %) and precision (about 75 %) is fairly good, but the sentence accuracy (about 25 %) is not. Schabes et al. (Schabes et al., 1993) reported their method resulted in 71.5% of brackets precision [3] and 6.8 % of sentence accuracy for test sentences whose length ranged from 20 to 30 words. The average length of our test sentences is about 23 words. Thus, our result is better than theirs. However, direct comparison is not very meaningful because the languages and the corpora dealt with are different.

6 Conclusion

In this paper, we proposed a method to extract a PCFG from a bracketed corpus. We also suggested two methods to improve the PCFG, one in removing redundant rules to reduce the grammar size, and the other in decreasing the number of parse trees. The latter is accomplished by introducing right recursive rules for compound words and subcategorizing the POSs. Finally, we conducted an experiment to evaluate the proposed methods. It shows that the accept rate is 94 %, the brackets recall is 85 %, the brackets precision is 75 % and the sentence accuracy is 25 %.

[3] They did not show the brackets recall.

One of the important problems to solve is that the extracted grammar still generates too many parse trees (10^9 on average). We need further improvement in order to prune unnecessary trees without degrading accuracies and coverage.

References

EDR, 1994. *EDR Electronic Dictionary User's Manual* (in Japanese). Japan Electronic Dictionary Research Institute, 2.1 edition.

M. Kiyono and J. Tsujii. 1994a. Combination of symbolic and statistical approaches for grammatical knowledge acquisition. In *Proceedings of the Conference on Applied Natural Language Processing*, pages 72–77.

M. Kiyono and J. Tsujii. 1994b. Hypothesis selection in grammar acquisition. In *Proceedings of the 14th International Conference on Computational Linguistics*, volume 2, pages 837–841.

Y. Kobayashi, T. Tokunaga, and H. Tanaka. 1994. Analysis of Japanese compound nouns using collocational information. In *Proceedings of the 14th International Conference on Computational Linguistics*, volume 2, pages 865–869.

K. Lari and S.J. Young. 1990. The estimation of stochastic context-free grammars using the Inside-Outside algorithm. *Computer speech and languages*, 4:35–56.

F. Pereira and Y. Schabes. 1992. Inside-Outside reestimation from partially bracketed corpora. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 128–135.

Y. Schabes, M. Roth, and R. Osborne. 1993. Parsing the Wall Street Journal with the Inside-Outside algorithm. In *Proceedings of the 6th Conference of European Chapter of the Association for Computational Linguistics*, pages 341–347.

K. Shirai, T. Tokunaga, and H. Tanaka. 1995. Automatic extraction of probabilistic context free grammar from a bracketed corpus (in Japanese). In *Proceedings of the 50th Annual Convention IPS Japan*, volume 3, pages 61–62.

M. Tomita. 1986. *An Efficient Parsing for Natural Languages*. Kluwer, Boston, Mass.