Robust N-gram Based Syntactic Analysis Using Segmentation Words

Nobuo INUI **Yoshiyuki KOTANI**
Department of Computer Science
Tokyo University of Agriculture and Technology
2-24-16 Nakacho Koganei
Tokyo Japan
{nobu, kotani}@cc.tuat.ac.jp

Abstract

We describe an N-gram based syntactic analysis using a dependency grammar. Instead of generalizing syntactic rules, N-gram information of parts of speech is used to segment a sequence of words into two clauses. A special part of speech, called **segmentation word**, which corresponds to the beginning or end symbol of clauses is introduced to express a sentence structure. Segmentation words for each clause were learned using the hill climbing method and a small bracketed corpus. Experimental results for Japanese sentences showed that N-gram based syntactic parser achieved 72.2% recall, which is about the same level of performance as a probabilistic context-free grammar based parser with human-made language-dependent information.

1 Introduction

Almost all stochastic syntactic parsers are based on context free grammars (especially, probabilistic context free grammar) (Beil et al. (1999); Charniak (1997); Liu and Soo (1994)) or its extensions, e.g. HPSG (Kanayama et al. (1999)) or LFG (Bod and Kaplan (1998)). Several researchers have tried to acquire syntactic rules from corpus automatically (Zhou and Ren (1999); Chelba and Jelinek (1998); Shirai et al. (1997); Pereira and Shabes (1992)). There are several issues in automatic rule acquisition:

(1) Erroneous inputs,
(2) The size of the training corpus,
(3) The number of syntactic rules.

A morphological analyzer normally passes words to a syntactic parser together with morphological information, like parts of speech, inflection and so on. Though morphological parsers have achieved high performance, using statistical information gathered from large corpora, syntactic parsers must allow for erroneous information about words, especially word segmentation error and tagging error. The available corpora with syntactic information are usually smaller than the corpora with morphological information and are not sufficient to acquire syntactic rules directly. In addition, many syntactic rules consisting of parts of speech and words are generated. It is necessary to generalize rules to process various sentences by sacrificing the performance. Human-made syntactic rules are only a portion of the rules needed in such a grammar. We think that information in the morphological corpus should be used for syntactic analysis.

Introducing language-dependent rules is a key to solving the above issues. But if we add such ad hoc rules, the performance of parser would improve only for the sentences based on which the rules were derived. We think that a parser should guarantee that it would process all sentences. To achieve this, the grammar model should not be constructed from a priori knowledge alone.

This paper proposes a method of handling these issues by using N-gram information of word or part of speech:

(1) N-gram information is collected from the corpus, which is generated by a morphological parser. Tagging errors and word segmentation errors are accounted in the N-gram information.
(2) The corpus is large enough for N-gram information to be very reliable.
(3) N-gram information calculated using a linear interpolation method which estimates N-gram information from under N-gram information approximates occurrence probabilities of various sentences.

To apply N-gram information to syntactic analysis, we describe here a model of syntactic analysis, some approximations of real world information and experimental results.

2 N-gram Based Model of Syntax

2.1 Formal Description of the Binary Dependency Grammar

For free-order languages like Japanese, dependency analysis, which focuses on modification relationships between phrases, is more suitable for the syntactic analysis. Unlike a context-free grammar, usually, a dependency grammar does not define non-terminal symbols like noun clauses, verb clauses and so on. The dependency grammar we use is formally denoted as shown below:

$$S(s) = \{m(p_i, p_j) \mid a\ clause\ p_i\ modifies\ p_j,\ p_i\ is\ a\ sequence\ of\ words\ or\ a\ word\} \cdots (1)$$

The following is an example of a dependency structure.

(Ex.1)
$$S(time\_flies\_like\_an\_arrow) = \{m(an, arrow), m(an\_arrow, like), m(like\_an\_arrow, flies), m(time, flies)\}$$

A dependency grammar cannot specify the order of words and generates many possible parse trees. To reduce the number of parse trees, the following principles (Nagao (1996)) are usually assumed in Japanese:

(1) Backward Referent: A modificand follows a modifier.
(2) No Crossing: Modification relations are not allowed to cross each other.
(3) Uniqueness of Modification: A modifier can modify only one word or clause.
(4) The Nearest Referent: A modifier modifies the nearest modificand.
(5) One Case for One Sentence: A verb takes restricted cases, which occur at most once.

Based on the above principles, we use a binary dependency grammar defined as follows:

$$S(s) = \{m(p_i, p_j \in p_{i+1}) \mid a\ clause\ p_i\ modifies\ a\ clause\ p_j\ that\ is\ in\ the\ next\ clause\ p_{i+1}\} \cdots (2)$$

Since a binary dependency grammar expresses only the relations between clauses which are side by side, the above principles are almost satisfied. For example, *p1* is a modifier and *p2* is a modificand in *m(p1,p2)* from principle (1). Though, in this case, a head word is in *p2*, we only solve modifier-modificand relation. In addition, since syntactic structures are usually denoted by a tree, they can easily be converted to binary dependency expressions. The difference between a binary dependency tree and the Chomsky normal form, which is a kind of binary context-free grammar, is the existence of non-terminal symbols. Consider a sequence of words 'I like it'. The Chomsky normal form assigns 'I' to *N*, 'like' to *V*, 'it' to *N*, then 'like it' to *VP* and, finally, 'I like it' to *S*. In this case, a

parser knows '*S->N VP*' and '*VP->V N*'[1]. In a binary dependency grammar, on the other hand, 'I' modifies 'like_it', i.e. m('I', 'like_it').

The following is an example of applying a binary dependency structure in Japanese.

(Ex. 2) Kyou wa yoku hareta ichinichi desu
today (subject) very fine all day be (It is very fine all day today)

(1) Expression like the context-free grammar (bracketed structure)
(((kyou) (wa) ((((yoku) (hareta)) (ichinichi)) (desu))

$$S(s) = \begin{Bmatrix} m(kyou,wa), m(yoku,hareta), m(yoku\_hareta, ichinichi), \\ m(yoku\_hareta\_ichinichi, desu), m(kyou\_wa\_yoku\_hareta\_ichinichi, desu) \end{Bmatrix}$$

(2) Expression using a traditional dependency grammar
(kyou wa) (yoku) (hareta) (ichinichi) (desu)

$$S(s) = \begin{Bmatrix} m(kyou,wa), m(yoku,hareta), m(ichinichi, desu), \\ m(hareta, ichinichi \in ichinichi\_desu), \\ m(kyou\_wa, desu \in yoku\_hareta\_ichinichi\_desu) \end{Bmatrix}$$

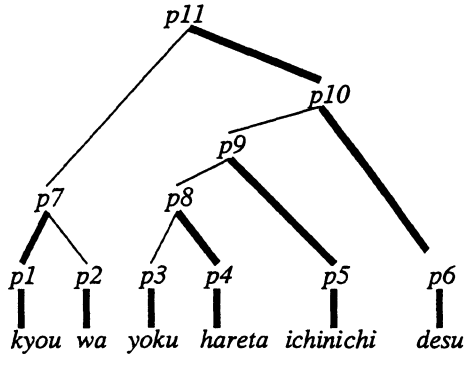


Fig.1 The binary dependency structure
pi means a clause or word. Thick lines show heads of phrase.

$$m(p1, p2) = p7, m(p3, p4) = p8, m(p8, p5) = p9, m(p9, p6) = p10, m(p7, p10) = p11$$

There are units, called "phrases", which are the minimum meaningful clauses in a traditional dependency grammar. In Japanese, we do not divide a noun "kyou" and a particle "wa", because "wa" cannot be used alone. A traditional dependency grammar forms sentence structures from phrases. For the above expression, a binary dependency structure can express both a context-free grammar style and a traditional dependency grammar style. The model shown in (2) was often used in previous research in which a decision tree was used to estimate a modification probability. In this paper, we use expression (1) in ex.2 which EDR corpus adopts, though our method is defined apart from structures of sentences.

2.2 A Probability Model of the Binary Dependency Grammar

The issue of finding an optimal structure can be identical with finding the most probable structure. We use the expression described below:

---

[1] Of course, there are variations of rules in the Chomsky normal form. For example, each rule might include two words, as in '*V1*->like it' and '*S1*->I *V1*'. In this case, a Chomsky normal form is the same as a binary dependency grammar, if the left-hand side of rule is uniquly determined by its right-hand side and the non-terminal symbols are different from each other when the right-hand sides of rules are different.

Probability of optimal segmentation $w_1 \cdots w_n =$

$$\arg\max_{l=1,\cdots,n-1} P(p_k)P(p_l)P(m(p_k, p_j \in p_l) \mid p_k, p_l) \quad \cdots(3)$$

where $p_k = w_1 \cdots w_i, p_l = w_{i+1} \cdots w_n$

The above expression contains two probabilities, the phrase occurrence probability $P(p_k)P(p_l)$ and the modification probability $P(m(p_k, p_j \in p_l) \mid p_k, p_l)$. The phrase occurrence probability reflects the strength of clauses. Ex.3 shows one way of segmenting a sentence. The syntactic parser finds the strongest clause and creates a syntactic structure. If longer N-gram information is available, it is sufficient to count only the modification probability, because it can express both the modification relation and the phrase occurrence. But we assume that both probabilities are required for syntactic parse when bigram or trigram information is used.

The EDR corpus (EDR (1996)) uses a context-free grammar-like structure. With this expression, both a word and a clause with a modifier must be explicitly displayed to express a sentence's structure. A traditional dependency grammar only shows the modified clause in its output. The context-free grammar-like structure expresses the original word order. In using a context-free grammar-like structure, the modification probability which means that a modifier modifies a modificand word or clause must be shown. To calculate this probability, our parser searches the tree structure of a sentence to find the optimal modificand. For example, the clause "kyou wa" directly modifies the clause "ichinichi desu" in the clause "yoku hareta ichinichi desu" in ex.3. If a sufficiently large corpus is available, the above two probabilities can be directly calculated as follows:

(Ex.3) kyou wa yoku hareta ichinichi desu
p1 *p2* *p3* *p4* *p5* *p6*
today (Subj) very fine all day be

yoku hareta	(yoku hareta)
yoku hareta ichinichi	((yoku hareta) ichinichi)
yoku hareta ichinichi desu	(((yoku hareta) ichinichi) desu)
kyou wa yoku hareta ichinichi desu	((kyou wa) (((yoku hareta) ichinichi) desu))

Constraints:

$P(p3p4), P(p1p2) > P(p2p3), P(p4p5), P(p5, p6)$

$P(p3p4p5) > P(p1p2p3p4), P(p5p6)$

$P(p3p4p5p6) > P(p1p2p3p4p5)$

$$P(p_i) = \frac{\textit{the number of a clause } p_i}{\textit{the total number of clause}} \quad \cdots(4)$$

$$P(m(p_i, p_j \in p_{i+1}) \mid p_i, p_j) = \frac{\textit{the number of } p_i \textit{ modifing } p_j \textit{ in } p_{i+1}}{(\textit{the number of clause } p_i)(\textit{the number of clause } p_j)} \cdots(5)$$

2.3 N-gram approximation of probabilities

It is hard to calculate the phrase occurrence probability and the modification probability from the corpus directly without overgeneralization or language-dependent information inputted by humans, since there are many phrases and relationships between clauses not recorded in a given corpus. To avoid this problem, we use N-gram information gathered by calculating a tagged corpus. To calculate the approximate phrase occurrence probability $P(p_i)$, we use the following expression using N-gram information.

$$P(p_i = w_1 w_2 \cdots w_n) \equiv P(w_0 w_1 w_2 \cdots w_n w_{n+1}) = \prod_{i=0}^{n+1} P(w_i \mid w_1 \cdots w_{i-1})$$
$$= P(w_1 \mid w_0) P(w_2 \mid w_0 \cdots P(w_{n+1} \mid w_0 w_1 \cdots w_n) \quad \cdots(6)$$

where w_0, w_{n+1} are special symbols denoting the beginning and the end of a clause, respectively.

Two special symbols, w_0 and w_{n+1} are used to express the phrase occurrence probability. Since we plan to acquire this probability from a tagged corpus, these two symbols are needed to show the strength of phrases. These symbols play an important role in expressing the sentence structure. Intuitively, **the beginning of a sentence and the end of a sentence seem to correspond to w_0 and w_{n+1}, respectively**. We use w_0 and w_{n+1} for calculating the strength of phrases described in the previous section. We call these special symbols **segmentation words**. For example, consider a sequence of words w_2w_3 in $w_1w_2w_3$. The frequency of w_2w_3 must be more than that of $w_1w_2w_3$ in tagged corpus, i.e. $P(w_2w_3)>P(w_1w_2w_3)$. If we choose a pair of segmentation words for these two phrases carefully, we can change these probabilities to become $P(w_2w_3)<P(w_1w_2w_3)$, In Japanese, particles seem to be w_0. For example, we can recognize the end of a clause, i.e. the next word is the beginning of the next clause, by "wa". A verb also seems w_{n+1}. The previous word of "desu" is usually the end of a clause". What is optimal segmentation word for a clause is not clear, because segmentation words can vary the style of sentence structure. So, we must determine segmentation words according to clauses. A method for learning segmentation words is described in section 4.

For the modification probability between clauses, we also use an N-gram approximation under the assumption that modified words or clauses are frequently placed next (or previous) to a modificand clause. Using this assumption, the modification probability is calculated to be approximately:

$$P(m(p_i, p_j \in p_{i+1}) \mid p_i, p_{i+1}) = \max P(p_i p_k) \quad p_k \in p_{i+1} \quad \cdots(7)$$

The following is an example from Ex.3.

(Ex.4)
$$P(m(kyou\_wa, yoku\_haretta\_ichinichi\_desu))$$
$$= \max(P(kyou\_wa, yoku), P(kyou\_wa, hareta), P(kyou\_wa, yoku\_hareta),$$
$$P(kyou\_wa, ichinichi), \cdots, P(kyou\_wa, yoku\_hareta\_ichinichi\_desu))$$

In general, a few words at the end of p_i and at the beginning of p_{i+1} are important to judge the modification relation. For example, the bigram information of "wa" and "desu" is important for m("kyou_wa", "desu") in ex.4. So we use the following expression to calculate the modification probability.

$$P(p_i p_k \mid p_i, p_k) = \frac{P(w_{ip} \cdots w_{in} w_{k1} \cdots w_{kp})}{P(w_{ip} \cdots w_{in}) P(w_{k1} \cdots w_{kq})}$$
$$p_i = w_{i1} \cdots w_{in}, p_k = p_{k1} \cdots p_{km} \quad 1 \le p \le n, 1 \le q \le m \quad \cdots(8)$$

3 Syntactic Analysis

We use the CKY algorithm that was developed for the Chomsky normal form to parse a sentence. Unlike a context-free grammar, it is not necessary for a parser to keep information about non-terminal symbols. Instead of non-terminal symbols, we extract a typical word in the modificand phrase. But this information is dependent on the structure of a sentence. For example, we showed two types of structures in Ex.2 in the last section. A typical word places the last phrase in structure (1), not in structure (2). To avoid this problem, we use the structural distance between a modificand phrase and a typical word in a modified phrase. This algorithm is described in Algorithm 1.

Algorithm 1 checks all possibilities of segmenting a sequence of words into two parts. The segmentation with the highest probability is a solution of the parse in each sequence of words.

```
s = w1w2w3...wn
Initialize the phrase occurrence matrix pom[i][j]=P(wi...wj) 1<=i<n i<=j<=n
Initialize the modification probability matrix mpm[i][j]=P(wi-1wiwjwj+1) 1<=i<=n-1, i<j<=n
Initialize the probability matrix pm[i][j]=0 1<=i<n, i<=j<=n
for(k=2; k<=n; k++)
    for(i=1; i<=n-k; i++)
        for(j=0; j<i-1; j++) {
          prob=pom[i][i+j]*pom[i+j+1][i+k+1]*max_mpm(i+j, i+j+1, i+k+1);
          if(pm[i][i+k+1]<prob) {
                pm[i][i+k+1]=prob;
                separating_point[i][i+k+1]=i+j;
          }
```

Algorithm 1. Bottom Up Parsing Algrorithm

The matrix element *pom[i][j]* contains the phrase occurrence probability for a phrase candidate $w_i...w_j$ with segmentation words. A function *max_mpm()* searches for a phrase in $w_{i+j+1}...w_{i+k+1}$ which is modified by the phrase $w_i...w_{i+j}$, recursively. This function finds the most plausible modificand and returns the modification probability between two clauses. This function is needed, since the entire next clause is not always a modificand. It is defined in Algorithm 2.

```
A function max_mpm(pre_end, post_beg, post_end)
{
  if(post_beg==post_end) return(mpm[pre_end][post_beg]);
  prob1=alpha*max_mpm(pre_end, post_beg,post_beg+separating_point[post_beg][post_end]);
  prob2=alpha*max_mpm(pre_end,post_beg+separating_point[post_beg][post_end]+1,post_end);
  return(max(prob1,prob2));
}
```

Algorithm 2 Find the Modificand Candidate with the Maximum Probability

In the function *max_mpm()*, the structural distance between phrases is used to weigh probabilities. A parameter *alpha* is heuristically determined for discounting ratio by the depth of a parsing tree. The modification probability is calculated using N-gram information, not the modification structure in our experiments.

4 Learning Segmentation Words

How segmentation words are determined is important in calculating the phrase occurrence probabilities. Generally, the beginning of a sentence and the end of a sentence become segmentation words explicitly as special symbols, but this is not an optimal solution. In English, a preposition which is next to a noun or a verb is also a segmentation key. It seems that the relation between a verb and a preposition is stronger than between a noun and a preposition. Therefore, the possibility of generating V+ADV_P is higher than that of generating N+ADV_P. This is a language-dependent problem. We use a small corpus to determine segmentation words for each word. Unfortunately, it is not known how to assign segmentation words. Though all possibilities for the probability of a segmentation word should be checked, it is impossible to do so because of computational complexity. Instead of the optimal probability, we try to assign a specific part of speech to a segmentation word for each word. The sub-optimal assignment can be found using the hill climbing method described in Algorithm 3.

An element of *beginning[i]* and *end[i]* is the beginning and end of a clause, i.e. segmentation words for each part of speech, which begins with a part of speech *i*, respectively. The hill climbing method tries to find the segmentation words which improve the performance of parsing. In this case, the hill

climbing method usually reaches a local maximum solution. To avoid this problem, (but not perfectly), we first arrange the order of checked words randomly. Experimentally, this method is very effective. The hill climbing method finds an optimal solution with slightly changed parameters. In our method, one of *beginning[i]* or *end[i]* is changed before it is checked.

5 Experimental Results

Fig. 2 illustrates the experimental set up for evaluating our system. We use a morphological analyzer (Inui and Kotani (1999)) to assign a part of speech to each word. The performance of this analyzer is estimated to be about 98% recall and precision. Since this analyzer was constructed using tagged corpora (Toyoura et.al. (1996)), it prefers to segment shorter words than the words in bracketed corpora (EDR (1996)). We use the bigram and trigram information for the phrase occurrence probability and the modification probability. The following interpolated expressions are used for them based on equations (6) and (8):

$$P(p_i) = P(w_1 \cdots w_n) \cong P(w_o)P(w_1 \mid w_0)P_{\text{int}}(w_2 \mid w_0\, w_1) \cdots P_{\text{int}}(w_{n+1} \mid w_{n-1} w_n)$$
$$\text{where} \quad P_{\text{int}}(w_i \mid w_{i-2} w_{i-1}) = \lambda P(w_i \mid w_{i-2} w_{i-1}) + (1-\lambda) P(w_i \mid w_{i-1}) \qquad \cdots (9)$$
$$0 \le \lambda \le 1$$

$$P(p_i p_k \mid p_i, p_k) = P((w_1 \cdots w_i)(w_{i+1} \cdots w_n) \mid (w_1 \cdots w_i), (w_{i+1} \cdots w_n))$$
$$\cong \frac{\lambda}{2}(P(w_{i-1} w_i w_{i+1} \mid w_{i-1} w_i, w_{i+1}) + P(w_i w_{i+1} w_{i+2} \mid w_i, w_{i+1} w_{i+2}))$$
$$+ (1-\lambda) P(w_i w_{i+1} \mid w_i, w_{i+1})$$
$$\text{where} \quad P(w_{i-1} w_i w_{i+1} \mid w_{i-1} w_i, w_{i+1}) = \frac{N(w_{i-1} w_i w_{i+1})}{N(w_{i-1} w_i) N(w_{i+1})} \quad (N(p_i) \text{ is the number of} \qquad \cdots (10)$$
$$0 \le \lambda \le 1 \qquad \text{a phrase in tagged corpus})$$

To determine segmentation words for each word is an ideal goal but somewhat unrealistic. So, instead of words, we use parts of speech. RWC corpus's classification of parts of speech (Toyoura et.al. (1996)) was used for our experiment. The following corpus is used to evaluate our method.

The bigram and trigram information: Gathered from RWC newspaper articles (**55,565,314 words**)
The number of parts of speech: **456**
Training corpus: EDR bracketed corpus, **100, 500, 1000 sentences**
(These sentences are used only to find segmentation words)
Testing corpus: EDR bracketed corpus **1000 sentences**

As the modification probability, we use a sequence of the last two words of the previous phrase and the first two words of the next phrase, since the statistics of the exact sequences of phrases is too sparse to use. The reason we use the last words and the first words is that it is a good approximation of a phrase in Japanese. Usually, we can judge the modification relation from them. We feel that the same can be done in English.

We use only 1,000 sentences to find the optimal segmentation words. A reason for this is that the hill climbing method is too slow to learn. This method checks all combination of segmentation words (456×456) in one generation cycle which is a period of the while-loop in algorithm 3. But experimental results (graph 2) show that the number of these sentences is not such a big problem.

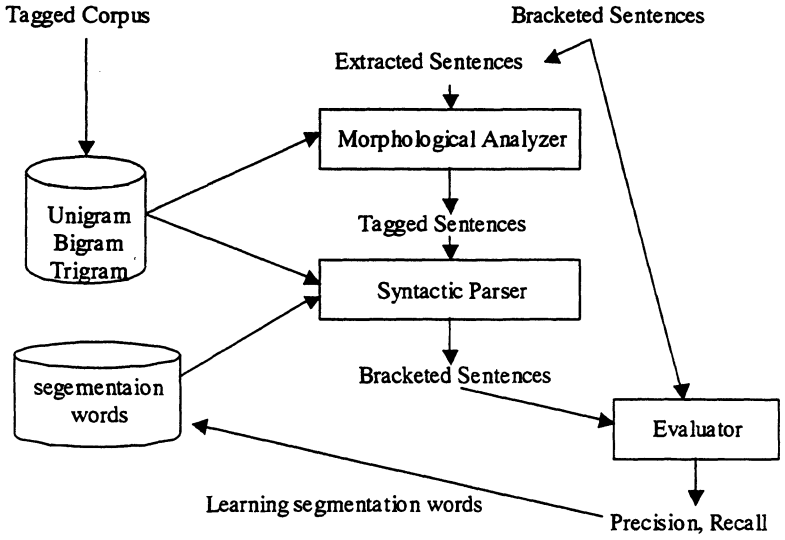
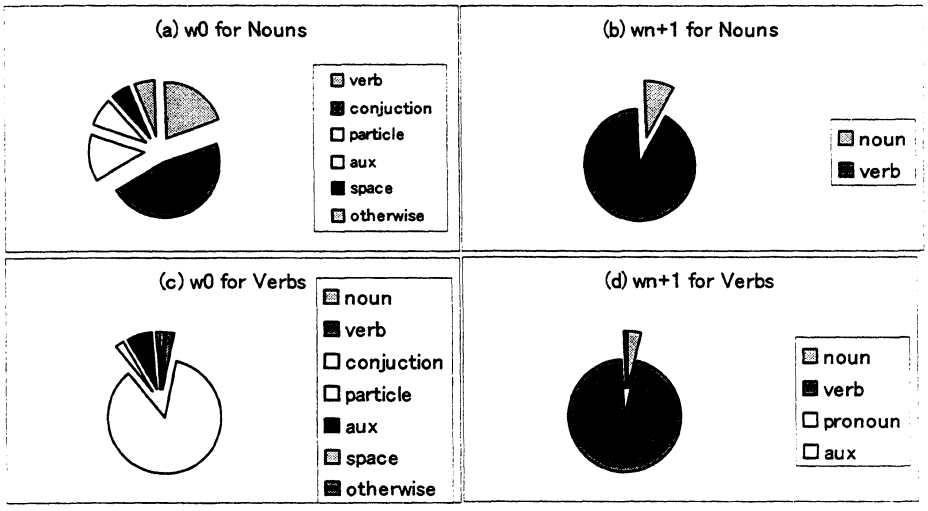


Fig. 2 Evaluation System

n: the kind of words
Initialize beginning*[i], end[i] 0<i<n* to a part of speech, appropriately.
Initialize *maximum_probability=0*
while(*maximum_probability is updated)* { // increment generation
  Initialize *randam_order_of_word random[i] 0<i<=2*n*
  for(*i*=1; *i*<=2**n*; *i*++)
    if(*randam[i]* is the beginning for a word *j*) {
      maximum_pos=beginning[j];
      for(*k*=1; *k*<=the number of part of speech; *k*++) {
        beginning[j]=k;
        prob=evaluation_of_small_corpus();
        if(maximum_prob<prob) {
          Update *maximum_probability* to *prob*
          maximum_pos=k;
        }
      }
      beginning[j]=maximum_pos;
    } else {
      maximum_pos=end[j];
      for(*k*=1; *k*<=*the number of part of speeh; k*++) {
        end[j]=k;
        prob=evaluation_of_small_corpus();
        if(*maximum_prob<prob*) {
          Update *maximum_probability* to *prob*
          maximum_pos=k;
        }
      }
      end[j]=maximum_pos;
    }
}
Algorithm 3 The Hill Climbing Method for Finding Optimal Segmentation Words

Graph 2 Segmentation Words (examples)



6 Discussion

The performance of our method may not seem so high, compared with other methods that reported over 80% recall and used hand-made syntactic rules. For example, Shirai et.al.'s (1997) system used EDR corpus to learn syntactic rules, and reported 62.71% recall. To generalize the rules acquired from EDR corpus directly, they introduced some heuristics dependent on Japanese. In addition, besides the part of speech information they used word information. On the other hand, our system uses only part of speech information. We believe our method has a potential to be equivalent to a context-free grammar based system.

Collins (1996) proposed an N-gram based parser. He used Treebank to extract N-gram information and the modification probability. We feel that the size of the available syntactic corpora is too small to gather N-gram information. Our method would be improved if syntactic training data were used. But our experimental results show that segmentation words can be used as a substitute for syntactic information.

It is important to note that no language-oriented rules are introduced into our system. Our system can analyze a sequence of words and generate a syntactic tree only from N-gram statistics. The only assumption we made was that words situated side by side have relationships. Applying this hypothesis to syntactic analysis, a system with no context-free grammar rules, can analyze sentences.

Selecting segmentation words is a key of our system. Our experiment shows that the system cannot acquire the optimal set of segmentation words from a small set of training sentences. It seems that, as while learning context-free grammar, the optimal set of segmentation words describes the training sentences only approximately. The existing part of speech information is used as a substitution for segmentation words in our experiments. Optimal segmentation words might be apart from the existing parts of speech. Instead of the hill climbing method, we conjecture that a genetic algorithm might improve the performance.

A part of speech system is not directly related to the syntactic analysis. Finding new word categories for syntactic analysis is a challenging issue. Our previous research (Inui and Kotani (1999)) has shown that an intermediate class of word categories, which exists between parts of speech and words was effective for N-gram based morphological analysis. We conjecture that new categories of words, like discourse information (Nasukawa (1995)), would improve our system.

Table 1 The size of data sets

	Traning	Testing
Sentences	1,000	1,000
Clauses	37,306	38,270
Words	24,312	23,328

Two factors, recall and precision, are used to evaluate the performance of the parser, similar to Shirai et al(1997). These are defined as followings.

$$Recall = \frac{the\ number\ of\ correct\ clauses}{the\ number\ of\ clauses\ in\ a\ data\ set} \cdots (11)$$

$$Precision = \frac{the\ number\ of\ correct\ clauses}{the\ number\ of\ clauses\ in\ proposed\ trees} \cdots (12)$$

In this context, a clause means a sequence of words. In the case that the output bracketed clause is different from the input bracketed clause, the result becomes correct, if the sequence of word is the same.

Fig.3 shows the learning curve for the hill climbing method described in the last section. In drawing this curve, the only phrase that consisted of more than two words was evaluated. Generation means the count of while-loop in Algorithm 3. From fig.3, when a generation proceeds, we can find that recall is going up and the hill climbing method for determining segmentation words is working. It seems that performance is improved when the size of the training data becomes large. Since the recall convergence is not so high, the hill climbing method seems to be in a local maximum solution.

The final result with 95% confidence limits is shown in Table 2. Precision is considerably lower than recall. This is because our morphological analyzer prefers segmenting words into short segments. To reduce the number of phrases in parsing trees, we can merge some words into one word as in the unknown word estimation.

Graph 2 shows rough classifications of segmentation words. There are 34 kinds of nouns and 381 kinds of verbs in RWC corpus. Nouns and verbs prefer conjunctions and verbs as w_0 and w_{n+1}, respectively. This graph shows that nouns require various w_0 to improve the parsing performance.

Graph 1 The Leaning Curves (Recall) for each Training Data Set

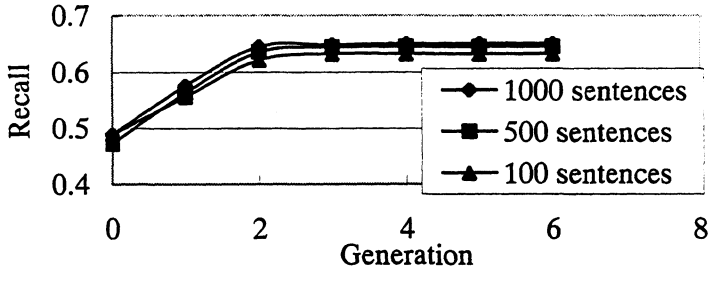


Table 2 Recall and Precision

Traning Sentences			
Recall	**LB95-UB95**	**Precision**	**LB95-UB95**
77.20%	76.6%-77.8%	60.7%	60.1%-61.2%
Testing Sentences			
Recall	**LB95-UB95**	**Precision**	**LB95-UB95**
72.90%	72.2%-73.5%	61.60%	61.0%-62.2%

7 Conclusions

A binary dependency grammar was introduced to describe syntactic structures and build a probabilistic model in this paper. The probabilistic model is composed of two probabilities, the phrase occurrence probability and the modification probability. A method of approximating both of these is introduced using N-gram statistics. The segmentation word, which describes the strength of a phrase, is learnt by the hill climbing method using an existing small bracketed corpus. We found that our method parsed sentences without language-dependent knowledge. This method is always successful in analyze a sentence, if the N-gram information is sufficient.

Acknowledgement

This study is supported by the ministry of education, science, sports and culture of Japan under grant in aid No.12780266. We would like to thank Professor Bipin Indurkhya of Tokyo University of Agriculture and Technology and Beryl Nelson for their advice and correcting of this paper.

References

Beil, F., Carrol, G., Prescher, D., Riezler, S. and Rooth, M. 1999. Inside-Outside Estimation of Lexicalized PCFG for German. *Proceedings of 37th ACL*: 269-266

Bod, R. and Kaplan, R. 1998. A Probabilistic Corpus-Driven Model for the Lexical-Functional Analysis. *Proceeding of the COLING-ACL'98*: 145-151.

Charniak, E. 1997. Statistical Parsing with a Context-free Grammar and Word Statistics. *Proceedings of AAAI'97*: 598-603.

Chelba, C. and Jelinek, F. 1998. Exploiting Syntactic Structure for Language Modeling. *Proceeding of the COLING-ACL'98*: 225-231.

Collins, M. 1996. A New Statistical Parser Based on Bigram Lexical Dependency. *Proceeding of the 34th ACL*: 184-191.

EDR. 1996. EDR Electric Dictionary Manual Ver. 1.5

Kanayama, H., Torisawa, K., Mitsuishi, Y. and Tsujii, J. 1999. Statistical Dependency Analysis with an HPSG-based Japanese Grammar. *Proceedings of 5th NLPRS*: 138-143.

Liu, R. and L., Soo, V. W. 1994. A Corpus-based Learning Techniques for Building a Self-Extensible Parser. *Proceedings of COLING'94*: 441-446

Inui, I. and Kotani Y. 1999. Finding the Best State for HMM Morphological Analyzer. *Proceeding of 5th NLPRS*: 44-49.

Nagao, M. (Ed.) 1996. SHIZENGENGOSYORI (Natural Language Processing), Iwanami Pub (In Japanese)

Nasukawa, T. 1995. Robust Parsing Based on Discourse Information. *Proceedings of 33rd ACL*: 39-46

Pereira F. and Shabes, Y. 1992. Inside-Outside Reestimation from Partially Bracketed Corpora. *Proceedings of the 30th ACL*: 128-135

Shirai, K., Tokunaga, T. and Tanaka H. 1997. Automatic Extraction of Japanese Probabilistic Context Free Grammar From a Bracketed Corpus. *Journal of Natural Language Processing*, 4(1): 125-146. (In Japanese)

Toyoura, J., Tokunaga, T. and Isahara, H. 1996. Development of RWC Text Database Tagged with Classification Code. IPSJ Technical Report, NL-114-5 (In Japanese)

Zhou, Q. and Ren, F. 1999. Automatic Inference for Chinese Probabilistic Context-Free Grammar. *Proceeding of 5th NLPRS*: 73-78.