A POLYNOMIAL-ORDER ALGORITHM FOR OPTIMAL PHRASE SEQUENCE SELECTION FROM A PHRASE LATTICE AND ITS PARALLEL LAYERED IMPLEMENTATION

Kazuhiko OZEKI
The University of Electro-Communications
Chofu, Tokyo, 182, Japan

Abstract

This paper deals with a problem of selecting an optimal phrase sequence from a phrase lattice, which is often encountered in language processing such as word processing and post-processing for speech recognition. The problem is formulated as one of combinatorial optimization, and a polynomial order algorithm is derived. This algorithm finds an optimal phrase sequence and its dependency structure simultaneously, and is therefore particularly suited for an interface between speech recognition and various language processing. What the algorithm does is numerical optimization rather than symbolic operation unlike conventional parsers. A parallel and layered structure to implement the algorithm is also presented. Although the language taken up here is Japanese, the algorithm can be extended to cover a wider family of languages.

1. Introduction

In Japanese language processing related to speech recognition and word processing, we often encounter a problem of selecting a phrase sequence which constitutes the most acceptable sentence from a phrase lattice, that is, a set of phrases with various starting and ending positions. By solving this problem, linguistic ambiguities and/or uncertainties coming from the inaccuracy in speech recognition are expected to be resolved.

This problem can be solved, in principle, by enumerating all the possible combinations of the phrases and measuring the syntactic and semantic acceptability of each phrase sequence as a sentence. Obviously, however, the amount of computation in this enumerative method grows exponentially with respect to the length of the sequence and becomes intractable even for a moderate problem size.

In this paper we formulate this task as a combinatorial optimization problem and derive a set of recurrence equations, which leads to an algorithm of polynomial order in time and space. We utilize the idea of dependency grammar [Hays 64] for defining the acceptability of a phrase sequence as a Japanese sentence.

With a review of recent theoretical development on this topic, a parallel and layered implementation of the algorithm is presented.

2. Dependency Structure of Japanese

In Japanese, words and morphemes are concatenated to form a linguistic unit called 'bunsetsu', which is referred to as simply 'phrase' here. A typical phrase consists of a content word followed by some functional morphemes. A Japanese sentence is a sequence of phrases with a structure which can be described by a diagram as in Fig.1 [Hashimoto 46]. For a sequence of phrases $x_1x_2\ldots x_n$ to be a well-formed Japanese sentence, it must have a structure satisfying the following constraints [Yoshida 72]:

(c1) For any i ($1\leq i\leq n-1$), there exists unique j ($i<j\leq n$) such that x_i modifies x_j in a wide sense.

(c2) For any i,j,k,l ($1\leq i<j<k<l\leq n$), it never occurs that x_i modifies x_k and x_j modifies x_l.

A structure satisfying these constraints is called a dependency structure here. More formally we define a dependency structure as follows [Ozeki 86a].

Definition 1

(1) If x_0 is a phrase, then $\langle x_0\rangle$ is a dependency structure.

(2) If $X_1,\ldots,X_n$ are dependency structures and x_0 is a phrase, then $\langle X_1\ldots X_n\ x_0\rangle$ is a dependency structure.

A dependency structure $\langle X_1\ldots X_n\ x_0\rangle$ ($X_i=\langle\ldots x_i\rangle$) implies that each x_i, which is the last phrase in X_i, modifies x_0. It is easily verified that a structure satisfying the constraints (c1) and (c2) is a dependency structure in the sense of Definition 1 and vice versa [Ozeki 86a].

When a dependency structure X is composed of phrases $x_1,x_2,\ldots,x_n$ we say that X is a dependency structure on $x_1x_2\ldots x_n$. The set of all the dependency structures on $x_1x_2\ldots x_n$ is denoted as $K(x_1x_2\ldots x_n)$; and for a sequence of phrase sets $A_1,A_2,\ldots,A_n$, we define

$$KB(A_1,A_2,\ldots,A_n)=\{X\mid X\in K(x_1x_2\ldots x_n),\ x_i\in A_i\ (1\leq i\leq n)\}.$$

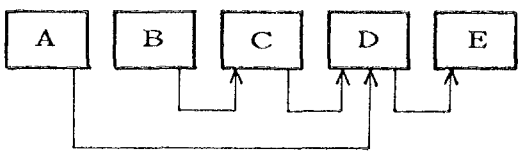

Fig.1 Example of dependency structure in Japanese. A,B,... are phrases.

3. Acceptability of a Dependency Structure

For a pair of phrases x_l and x_0, we can think of a penalty imposed on a modifier-modificant relation between x_l and x_0. This non-negative value is denoted as $pen(x_l;x_0)$. The smaller value of $pen(x_l;x_0)$ represents the more natural linguistic relation. Although it is very important to establish a way of computing $pen(x_l;x_0)$, we will not go into that problem in this paper. Based on the 'local' penalty, a 'global' penalty P(X) of a dependency structure X is defined recursively as follows [Ozeki 86a].

Definition 2

(1) For $X=\langle x\rangle$, $P(X)=0$.

(2) For $X=\langle X_1 \ldots X_n\ x_0\rangle$, where $X_i=\langle \ldots x_i\rangle$ $(1\leq i\leq n)$ is a dependency structure,

$$P(X)= P(X_1)+\ldots+P(X_n) +pen(x_1;x_0)+\ldots+pen(x_n;x_0).$$

Note that P(X) is the sum of the penalty of all the phrase pairs which are supposed to be in modifier-modificant relation in the dependency structure X. This function is invariant under permutation of $X_1,\ldots,X_n$ in accordance with the characteristic of Japanese.

4. Formulation of the Problem

For simplicity, let us begin with a special type of phrase lattice composed of a sequence of phrase sets $B_1,B_2,\ldots,B_N$ as shown in Fig.2, which we call phrase matrix. Suppose we are given a phrase matrix and a reliability function

$$s: B_1 \cup B_2 \cup \ldots \cup B_N \dashrightarrow R_+,$$

where R_+ denotes the set of non-negative real numbers. The smaller value of s(x) represents the higher reliability of x. We encounter this special type of phrase lattice in isolated phrase speech recognition. In that case B_i is the set of output candidates for the ith utterance, and s(x) is the recognition score for a candidate phrase x.

For a dependency structure X on a phrase sequence $x_1x_2\ldots x_N$, the total reliability of X is defined as

$$S(X)= s(x_1)+\ldots+s(x_N).$$

Combining the acceptability and the reliability, we define an objective function F(X) as

$$F(X)= P(X)+S(X).$$

B_1	B_2	•	•	•	B_N
x_{11}	x_{21}	-	-	-	x_{N1}
x_{12}	x_{22}	-	-	-	x_{N2}
x_{13}	x_{23}	-	-	-	x_{N3}
·	·				·
·	·				·
·	·				·

Fig.2 Phrase matrix. $B_1,\ldots,B_N$ are phrase sets.

Then the central problem here is formulated as the following combinatorial optimization problem [Matsunaga 86, Ozeki 86a].

Problem Find a dependency structure

$$X\in KB(B_1,B_2,\ldots,B_N)$$

which minimizes the objective function F(X).

By solving this problem, we can obtain the optimal phrase sequence and the optimal dependency structure on the sequence simultaneously.

When $|B_1|=|B_2|=\ldots=|B_N|= M$, we have

$$|KB(B_1,B_2,\ldots,B_N)|= ({}_{2(N-1)}C_{(N-1)})/N)M^N,$$

where C denotes combination. This becomes a huge number even for a moderate problem size, rendering an enumerative method practically impossible.

5. Recurrence equations and a resulting algorithm

Combining two dependency structures X and $Y=\langle Y_1,\ldots,Y_m,y\rangle$, a new dependency structure $\langle X,Y_1,\ldots,Y_m,y\rangle$ is obtained which is denoted as $X \oplus Y$. Conversely, any dependency structure Z with length greater than 1 can be decomposed as $Z= X \oplus Y$, where X is the top dependency structure in Z. Moreover, it is easily verified from the definition of the objective function that

$$F(Z)= F(X) + F(Y) + pen(x;y),$$

where x and y are the last phrases in X and Y, respectively. The following argument is based on this fact.

We denote elements in B_j as $x_{j1},x_{j2},\ldots$. For $1\leq i\leq j\leq N$ and $1\leq p\leq |B_j|$, where $|B_j|$ denotes the number of elements in B_j, we define

$$opt(i,j;p) =\min\{F(X)\,|\,X\in KB(B_i,\ldots,B_{j-1}\{x_{jp}\})\}$$

and

$$opts(i,j;p) =\operatorname{argmin}\{F(X)\,|\,X\in KB(B_i,\ldots,B_{j-1}\{x_{jp}\})\}.$$

Then the following recurrence equations hold for opt(i,j;p) and opts(i,j;p), respectively [Ozeki 86a].

Proposition 1 For $1\leq i\leq j\leq N$ and $1\leq p\leq |B_j|$,

(1) if i=j, then $opt(i,j;p)=s(x_{jp})$,

(2) and if i<j, then

$$opt(i,j;p) =\min\{f(k,q)\,|\,i\leq k\leq j-1, 1\leq q\leq |B_k|\},$$

where

$$f(k,q)=opt(i,k;q)+opt(k+1,j;p) +pen(x_{kq};x_{jp}).$$

Proposition 1' For $1\leq i\leq j\leq N$ and $1\leq p\leq |B_j|$,

(1) if i=j, then $opts(i,j;p)=\langle x_{jp}\rangle$,

(2) and if i<j, then

$$opts(i,j;p) =opts(i,*k;*q)\oplus opts(*k+1,j;p),$$

where *k is the best segmentation point and *q is the best phrase number in B_{*k}:

$$(*k,*q)=\operatorname{argmin}\{f(k,q)\,|\,i\leq k\leq j-1, 1\leq q\leq |B_k|\}.$$

According to Proposition 1, if the values of opt(i,k;q) and opt(k+1,j;p) are known for $1\leq k\leq j-1$ and $1\leq q\leq |B_k|$, it is possible to calculate the value of opt(i,j;p) by searching the best segmentation point and the best phrase number at the segmentation point. This fact enables us to calculate the value

of opt(1,N;p) recursively, starting with opt(i,i;q) ($1 \le i \le N, 1 \le q \le |B_i|$). This is the principle of dynamic programming [Bellman 57].

Let $*p = \text{argmin}\{\text{opt}(1,N;p) \mid 1 \le p \le |B_N|\}$, then we have the final solution

$\text{opt}(1,N;*p) = \min\{F(X) \mid X \varepsilon KB(B_1,\ldots,B_N)\}$

and

$\text{opts}(1,N;*p) = \text{argmin}\{F(X) \mid X \varepsilon KB(B_1,\ldots,B_N)\}$.

The opts(1,N;*p) can be calculated recursively using Proposition 2. Fig.3 illustrates an algorithm translated from these recurrence equations [Ozeki 86a]. This algorithm uses two tables, table1 and table2, of upper triangular matrix form as shown in Fig.4. The (i,j) element of the matrix has $|B_j|$ 'pigeon-holes'. The value of opt(i,j;p) is stored in table1 and the pair of the best segmentation point and the best phrase number is stored in table2. It should be noted that there is much freedom in the order of scanning i,j and p, which will be utilized when we discuss a parallel implementation of the algorithm.

```
Optimal_Dependency_Structure;
begin
/* Analysis Phase */
for j:=1 to N do
 for i:=j downto 1 do
  for p:=1 to |Bj| do
   if i=j then
    table1(i,j;p):=s(xjp);
   else
    begin
     table1(i,j;p)
     :=min{table1(i,k;q)+table1(k+1,j;p)
           +pen(xkq;xjp)
       |1≤k≤j-1,1≤q≤|Bk|};
     table2(i,j;p)
     :=argmin{table1(i,k;q)
              +table1(k+1,j;p)
              +pen(xkq;xjp)
       |1≤k≤j-1,1≤q≤|Bk|};
    end;
/* Composition Phase */
*p:=argmin{table1(1,N;p)
          |1≤p≤|BN|};
result:=opts(1,N;*p);
end.

function opts(i,j;p):char string;
begin
 if i=j then
  opts:='<xjp>';
 else
  begin
   (*k,*q):=table2(i,j;p);
   opts:=opts(i,*k;*q) ⊕ opts(*k+1,j;p);
  end;
end.
```

Fig.3 Algorithm to select an optimal dependency structure from a phrase matrix.

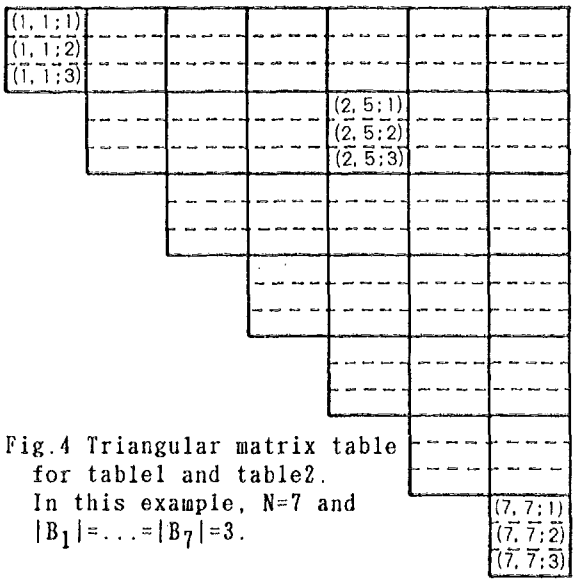

Fig.4 Triangular matrix table for table1 and table2. In this example, N=7 and $|B_1| = \ldots = |B_7| = 3$.

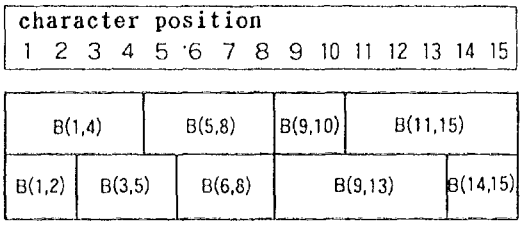

Fig.5 Example of phrase lattice.

When $|B_1| = \ldots = |B_N| = M$, the number of operations(additions and comparisons) necessary to fill table1 is $O(M^2N^3)$.

These recurrence equations and algorithm can be easily extended so that they can handle a general phrase lattice. A Phrase lattice is a set of phrase sets, which looks like Fig.5. B(i,j) denotes the set of phrases beginning at character position i and ending at j. A phrase lattice is obtained, for example, as the output of a continuous speech recognition system, and also as the result of a morphological analysis of non-segmented Japanese text spelled in kana characters only. We denote the elements of B(i,j) as $x_{ij1}, x_{ij2}, \ldots$, and in parallel with the definition of opt and opts, we define opt' and opts' as follows.

For $1 \le i \le m \le j \le N$ and x_{mjp},

opt'(i,j,m;p)

=the minimum value of [P(X)+S(X)] as X runs over all the dependency structures on all the possible phrase sequences beginning at i and ending at j with the last phrase being fixed as x_{mjp}.

and

opts'(i,j,m;p)

=the dependency structure which gives the above minimum.

Then recurrence equations similar to Proposition 1 and Proposition 1' hold for opt' and opts'[Ozeki 86b]:

Proposition 2 For $1 \le i \le m \le j \le N$ and $1 \le p \le |B(m,j)|$,

(1) if i=m, then $\text{opt}'(i,j,m;p) = s(x_{mjp})$,

(2) and if i<m, then
opt'(i,j,m;p)
=min{f'(k,n,q)|i≤n≤k≤m-1,1≤q≤|B(n,k)|},
where
f'(k,n,q)= opt'(i,k,n;q)+opt'(k+1,j,m;p)
+pen(x_{nkq};x_{mjp}).
Proposition 2' For 1≤i≤m≤j≤N and
1≤p≤|B(m,j)|,
(1) if i=m, then opts'(i,j,m;p)=<x_{mjp}>,
(2) and if i<m, then
opts'(i,j,m;p)
=opts'(i,*k,*n;*q) ⊕ opts'(*k+1,j,m;p),
where *k is the best segmentation point, *n is the top position of the best phrase at the segmentation point and *q is the best phrase number in B(*n,*k):
(*k,*n,*q)
=argmin{f(k,n,q)|i≤n≤k≤m-1,1≤q≤|B(n,k)|}.
The minimum is searched on 3 variables in this case. It is a straight forward matter to translate these recurrence equations into an algorithm similar to Fig.3 [Ozeki 86b, Kohda 86]. In this case, the order of amount of computation is $O(M^2N^5)$, where M=|B(i,j)| and N is the number of starting and ending positions of phrases in the lattice.

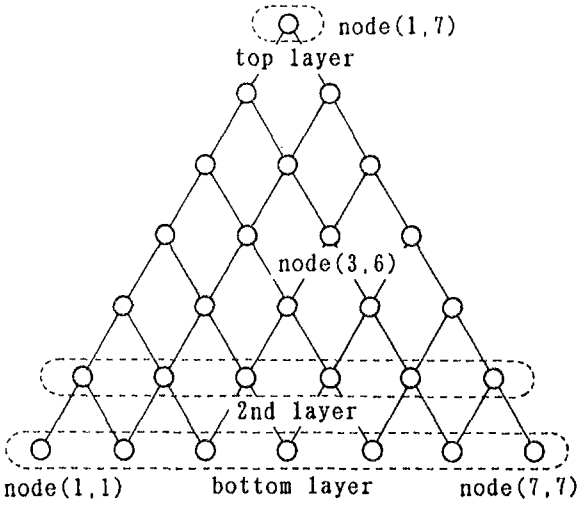

Fig.6 2-dimensional array of computing elements.

Also, we can modify the algorithm in such a way that up to kth optimal solutions are obtained.

6. Parallel and Layered Implementation

When only one processor is available, the amount of computation dominates the processing time. On the other hand, when there is no limit as to the number of processors, the processing time depends on how much of the computation can be executed in parallel. There exists a tidy parallel and layered structure to implement the above algorithm.

For simplicity, let us confine ourselves to a phrase matrix case here. Furthermore, let us first consider the case where there is only one element x_i in each of the phrase set B_i. If we define
opt''(i,j)=min{P(X)|X∈K(x_i,...,x_j)}
then Proposition 1 is reduced to the following simpler form.

Proposition 3 For 1≤i≤j≤N,
(1) if i=j, then opt''(i,j)=0,
(2) and if i<j, then
opt''(i,j)
=min{opt''(i,k)+opt''(k+1,j)
+pen(x_k;x_j)|i≤k≤j-1},

It is easy to see that opt''(i,j) and opt''(i+m,j+m) (m≠0) can be calculated independently of each other. This motivates us to devise a parallel and layered computation structure in which processing elements are arranged in a 2-dimensional array as shown in Fig.6. There are N(N+1)/2 processing elements in total. The node(i,j) has an internal structure as shown in Fig.7, and is connected with node(i,k) and node(k+1,j) (1≤k≤j-1) as in Fig.8. The bottom elements, node(i,i)'s (1≤i≤N), hold value 0 and do nothing else. The node(i,j) calculates the value of opt''(i,j) and holds the result in memory 1 together with the optimal segmentation point in memory 2. Within a layer all the nodes work independently in parallel and the computation proceeds from the lower to upper layer. An upper node receives information about a longer sub-sequence than a lower node: an upper node processes more global information than a lower node. When

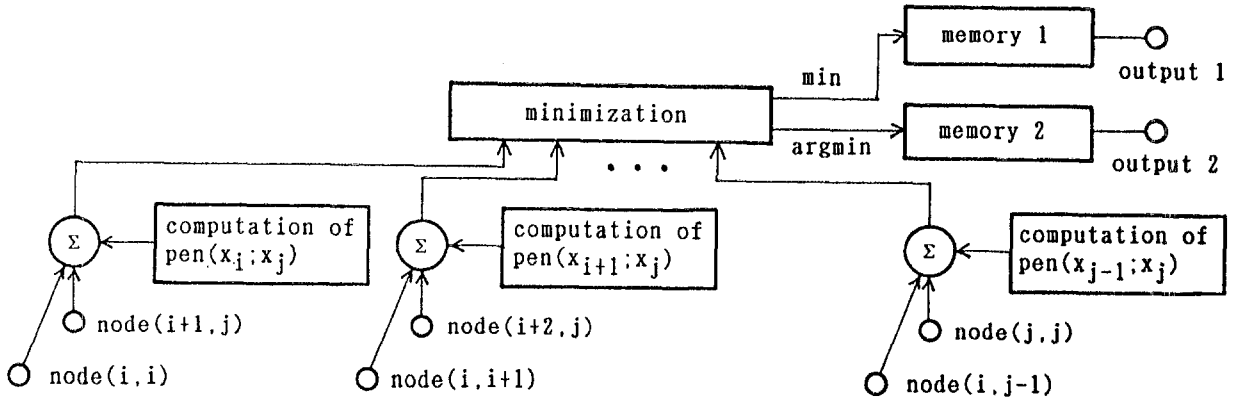

Fig.7 Internal structure of node(i,j).

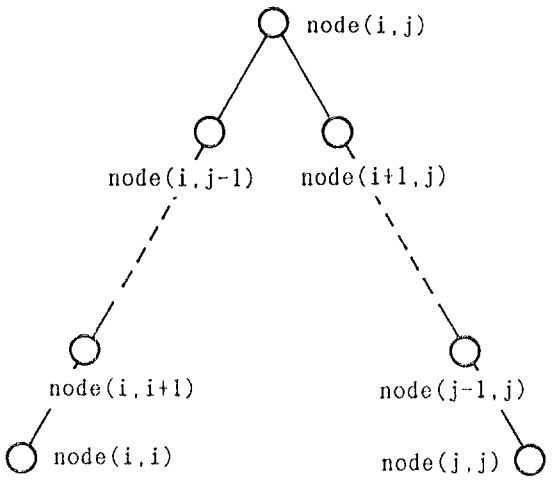

Fig.8 Nodes connected to node(i,j).

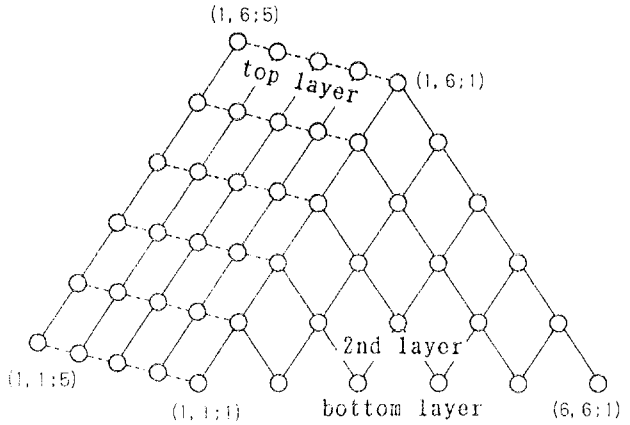

Fig.9 3-dimendional array of computing elements.

the top element, node(1,N), finishes its job, each node holds information which is necessary to compose the optimal dependency structure on $x_1 x_2 \ldots x_N$. This computation structure, having many simple inter-related computing elements, might be reminiscent of a connectionist model or a neural network.

This result can be easily extended, based on Proposition 1, to the case in which each phrase set has more than one elements. In this case processing elements are arranged in a 3-dimensional array as shown in Fig.9. The bottom elements, node(i,i;p)'s, hold the value of $s(x_{ip})$. The node(i,j;p) calculates the value of opt(i,j;p). The computation proceeds from the lower to upper layer just as in the previous simpler case. Further extension of this structure is also possible so that it can handle a general phrase lattice.

7. Related Works

The problem of selecting an appropriate phrase sequence from a phrase lattice has been treated in the field of Japanese word processing, where a non-segmented Japanese text spelled in kana character must be converted into an orthographic style spelled in kana and kanji. Several practical methods have been devised so far. Among them, the approach in [Oshima 86] is close in idea to the present one in that it utilizes the Japanese case grammar in order to disambiguate a phrase lattice. However, their method is enumeration-oriented and some kind of heuristic process is necessary to reduce the size of the phrase lattice before syntactic analysis is performed.

In order to disambiguate the result of speech recognition, an application of dependency analysis was attempted [Matsunaga 86, Matsunaga 87]. The algorithm used is a bottom-up, depth-first search, and it is reported that it takes considerable processing time. By introducing a beam search technique, computing time can be very much reduced [Nakagawa 87], but with loss of global optimality.

Perhaps the most closely related algorithm will be (extended)CYK algorithm with probabilistic rewriting rules [Levinson 85, Ney 87, Nakagawa 87]. In spite of the difference in the initial ideas and the formulations, both approaches lead to similar bottom-up, breadth-first algorithms based on the principle of dynamic programming.

In Fig.2, if each phrase set has only one phrase, and the value of between-phrase penalty is 0 or 1, then the algorithm reduces to the conventional Japanese dependency analyzer [Hitaka 80]. Thus, the algorithm presented here is a twofold extension of the conventional Japanese dependency analyzer: the value of between-phrase penalty can take an arbitrary real number and it can analyze not only a phrase sequence but a phrase matrix and a phrase lattice in polynomial time.

We have considered a special type of dependency structure in this paper, in which a modificant never precedes the modifier as is normally the case in Japanese. It has been shown that the algorithm can be extended to cover a more general dependency structure [Katoh 89].

The fundamental algorithm presented here has been modified and extended, and utilized for speech recognition [Matsunaga 88].

8. Concluding Remarks

In the method presented here, the linguistic data and the algorithm are completely separated. The linguistic data are condensed in the penalty function which measures the naturalness of modifier-modificant relation between two phrases. No heuristics has slipped into the algorithm. This makes the whole procedure very transparent.

The essential part of the algorithm is execution of numerical optimization rather than symbolic matching unlike conventional parsers. Therefore it can be easily implemented on an arithmetic processor such as DSP (Digital Signal Processor). The parallel

and layered structure will fit LSI implementation.

An obvious limitation of this method is that it takes account of only pair-wise relation between phrases. Because of this, the class of sentences which have a low penalty in the present criterion tends to be broader than the class of sentences which we normally consider acceptable. Nevertheless, this method is useful in reducing the number of candidates so that a more sophisticated linguistic analysis becomes possible within realistic computing time in a later stage.

A reasonable way of computing the penalty for a phrase pair is yet to be established. There seems to be two approaches to this problem: a deterministic approach taking syntactic and semantic relation between two phrases into consideration, and a statistical one based on the frequency of co-occurence of two phrases.

Acknowledgement

The author is grateful to the support of Hoso Bunka Foundation for this work.

References

[Bellman 57] Bellman,R.: 'Dynamic Programming', Princeton Univ. Press, 1957.

[Hashimoto 46] Hashimoto,S.: 'Kokugo-gaku Gairon', Iwanami, 1946.

[Hays 64] Hays,D.G.: 'Dependency Theory: A Formalism and Some Observations', Language, Vol.40, No.4, pp.511-525, 1964.

[Hitaka 80] Hitaka,T. and Yoshida,S.: 'A Syntax Parser Based on the Case Dependency and Its Efficiency', Proc. COLING'80, pp.15-20, 1980.

[Katoh 89] Katoh,N. and Ehara,T. : 'A fast algorithm for dependency structure analysis', Proc. 39th Annual Convention IPS Japan, 1989.

[Kohda 86] Kohda,M.: 'An algorithm for optimum selection of phrase sequence from phrase lattice',Paper Tech. Group, IECE Japan, SP86-72, pp.9-16, 1986.

[Levinson 85] Levinson,S.E.: 'Structural Methods in Automatic Speech Recognition', Proc. of IEEE, Vol.73, No.11, pp.1625-1649, 1985.

[Matsunaga 86] Matsunaga,S. and Kohda,M.: 'Post-processing using dependency structure of inter-phrases for speech recognition', Proc. Acoust. Soc. Jpn. Spring Meeting, pp.45-46, 1986.

[Matsunaga 87] Matsunaga,S. and Kohda,M.: 'Speech Recognition of Minimal Phrase Sequence Taking Account of Dependency Relationships between Minimal Phrases', Trans.IEICE Vol.J70-D,No.11, pp.2102-2107, 1987.

[Matsunaga 88] Matsunaga,S. and Kohda,M.: 'Linguistic processing using a dependency structure grammar for speech recognition and understanding', Proc. COLING'88, pp.402-407, 1988.

[Nakagawa 87] Nakagawa,S. and Ito,T. : 'Recognition of Spoken Japanese Sentences Using Mono-Syllable Units and Backward Kakari-Uke Parsing Algorithm', Trans. IEICE Vol.J70-D,No.12, pp.2469-2478, 1987.

[Nakagawa 87] Nakagawa,S : 'Unification of Kakari-Uke Analysis and Context-Free Parsing by CYK Algorithm for Continuous Speech Recognition', Proc. Acoust. Soc. Jpn. Spring Meeting, pp.131-132, 1987.

[Ney 87] Ney,H.: 'Dynamic Programming Speech Recognition Using a Context-Free Grammar', Proc. ICASSP'87, pp.69-72, 1987.

[Oshima 86] Oshima,Y., Abe,M., Yuura,K. and Takeichi,N. : 'A Disambiguation Method in Kana-Kanji Conversion Using Case Frame Grammar', Trans. IPSJ, Vol.27, No.7, pp.679-687, 1986.

[Ozeki 86a] Ozeki,K.: 'A multi-stage decision algorithm for optimum bunsetsu sequence selection', Paper Tech. Group, IECE Japan, SP86-32, pp.41-48, 1986.

[Ozeki 86b] Ozeki,K.: 'A multi-stage decision algorithm for optimum bunsetsu sequence selection from bunsetsu lattice', Paper Tech. Group, IECE Japan, COMP86-47, pp.47-57, 1986.

[Yoshida 72] Yoshida,S.: 'Syntax analysis of Japanese sentence based on kakariuke relation between two bunsetsu', Trans. IECE Japan, Vol.J55-D, No.4, 1972.