TR-I-0033

A Phoneme Lattice Parsing for Continuous Speech Recognition

音韻ラティスを用いての文節認識

Hiroaki Saito
斉藤 博昭

1988.7

概要

This paper describes an experiment in continuous speech recognition. The task is producing phrase candidates from a phoneme lattice. Because lattice parsing requires a much more extensive search than conventional string input parsing, an LR parsing algorithm is adopted as an efficient parsing method. The system is evaluated against continuous speech data using a fairly extensive context-free grammar. An 82 % recognition rate is achieved for the top five candidates. Average parsing time is 13 seconds.

ATR Interpreting Telephony Research Laboratories
ATR 自動翻訳電話研究所

1. Introduction

This paper describes an experiment in continuous speech recognition. The task is producing phrase candidates from a phoneme lattice. A phoneme lattice is a set of hypothesized phonemes with different time frames in the speech signal. An example of a phoneme lattice is shown in Figure 1-1.

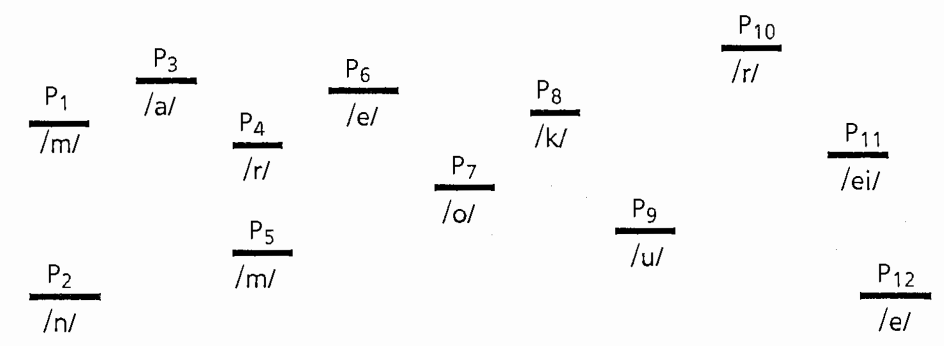

Figure 1-1: A phoneme lattice

Since lattice parsing requires a much more extensive search than conventional string input parsing, an efficient parsing algorithm has been adopted.

Section 2 describes the parsing algorithm and gives an example. Section 3 gives the grammar and data specifications for the task. The experiment result is given in Section 4. Finally, concluding remarks are given in Section 5.

2. Parsing Algorithm

The adopted parsing algorithm[5] is based on the LR parsing method in that the algorithm uses an LR parsing table constructed from a context-free grammar. The parsing proceeds from left to right with no backtracking. The LR parsing method was originally developed for compilers and has been extended for handling arbitrary context-free grammars[4].

2.1. A context-free grammar

A sample context-free grammar is shown in Figure 2-1.

```
-------------------------
(1) S    -->  NP V
(2) NP   -->  N
(3) NP   -->  N P
(4) N    -->  m a m e
(5) N    -->  a r e
(6) P    -->  o
(7) V    -->  o k u r e
(8) V    -->  k u r e
-------------------------
```

Figure 2-1: A sample grammar

Note that the grammar's terminal symbols are alphabetical characters, which can also be considered phonemes, instead of the usual grammatical categories such as verbs, nouns, prepositions, etc.

2.2. LR parsing table

The LR parsing table in Figure 2-2 is constructed automatically from the context-free grammar in Figure 2-1.

	a	u	e	o	k	m	r	$	S	N	V	P	NP
0	s2					s3			5	4			1
1				s7	s6						8		
2							s9						
3	s10												
4				s11,r2								12	
5								acc					
6		s13											
7					s14								
8								r1					
9			s15										
10						s16							
11				r6									
12				r3									
13							s17						
14		s18											
15				r5									
16			s19										
17			s20										
18							s21						
19				r4									
20								r8					
21			s22										
22								r7					

Figure 2-2: LR Parsing Table

Entries "s *n*" in the action table (the left part of the table) indicate the action: "shift one word from input buffer onto the stack and go to state *n*". Entries "r *n*" indicate the action: "reduce constituents on the stack using rule *n*". The entry "acc" stands for the action "accept", and blank spaces represent "error". The goto table (the right part of the table) decides which state the parser should go to after a reduce action. The terminal symbol $ represents the end of the input.

The LR parsing table in Figure 2-2 is different from the usual LR tables utilized by compilers of programming languages in that there is a multiple entry, called *conflicts,* on the row of state 4. While the encountered entry has only one action, parsing proceeds exactly the same way as normal LR parsing. When there are multiple actions in one entry, all the actions are executed in parallel. The adopted parsing algorithm can handle such multiple entries as shown in Section 2.4.

2.3. Phoneme Juncture Validation

In parsing the phoneme lattice, only certain phonemes can be connected to each other. For this phoneme juncture validation a predicate "Junct" is defined as follows.

Junct(PH_1,PH_2) returns TRUE if and only if the argument phonemes PH_1 and PH_2 can abut. When PH_1 is a starting symbol START, Junct(START, PH_2) returns TRUE if and only if PH_2 can be a starting phoneme. When PH_2 is an ending symbol $, Junct($PH_1$, $) returns TRUE if and only if PH_1 can end the speech signal.

2.4. Trace example

A trace of parsing the phoneme lattice in Figure 1-1 is shown here. The phonemes in Figure 1-1 are sorted by the ending frame number and are read one by one. The grammar in Figure 2-1 and the LR table in Figure 2-2 are used.

First, an initial state 0 is created. The LR table indicates that state 0 is expecting /a/ and /m/. This situation is described in Figure 2-3.

The first phoneme P_1(/m/) is expected by the initial state. On the assumption that Junct(START, P_1) returns TRUE, P_1 is connected to the initial state. The action table indicates that the next state number is 3, which is expecting /a/ (Figure 2-4).

The second phoneme /n/ is immediately rejected, because the action table is not expecting /n/.

The third phoneme /a/ is expected both by the initial state 0 and by the arc created by P_1. Assuming that both Junct(START, P_3) and Junct(P_1, P_3) return TRUE, P_3 is connected to both. Because the action table indicates that the shifted state numbers are different (ActionTable(0, a)=shift 2, ActionTable(3, a)=shift 10), two /a/ arcs are created for each. According to the action table, one /a/ arc with state 2 connected to the initial state is expecting /r/ and the other /a/ arc connected to the arc P_1 is expecting /m/(Figure 2-5).

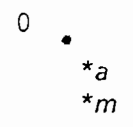

Figure 2-3

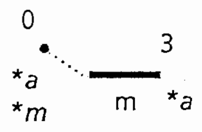

Figure 2-4

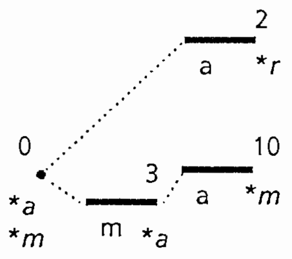

Figure 2-5

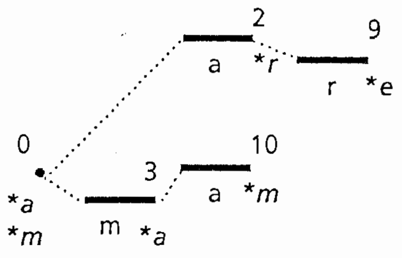

Figure 2-6

The fourth phoneme /r/ is expected by the arc P_3. Assuming that Junct(P_3, P_4) returns TRUE, the state number 9 is determined by the action table for the new arc which is expecting /e/(Figure 2-6).

The next phoneme P_5 /m/ is expected by the initial state and the P_3 arc. On the assumption that Junct(START, P_5) is *not* TRUE and Junct(P_3, P_5) is TRUE, P_5 is connected to the P_3 arc. The action table indicates that the state number is 16 and the arc is expecting /e/(Figure 2-7).

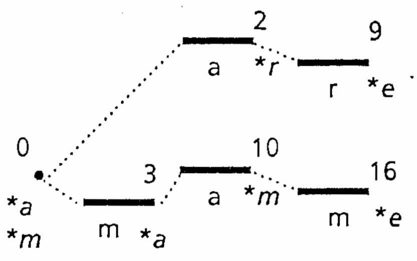

Figure 2-7

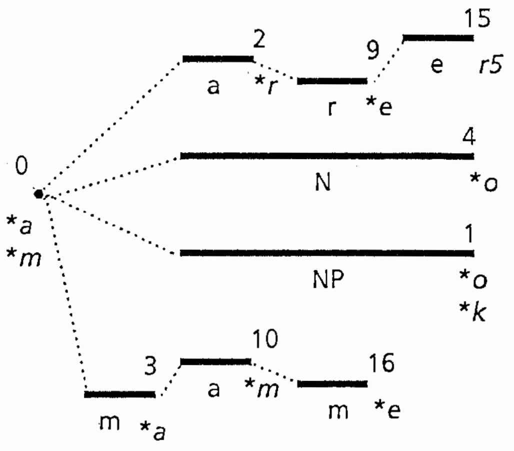

Figure 2-8

The next phoneme P_6 (/e/) is expected by two arcs P_4 and P_5. It is assumed that both Junct(P_4, P_6) and Junct(P_5, P_6) are TRUE. Connected to P_4, the state number is 15 according to the action table. Its state is ready for the reduce action (reduce 5). Thus using the rule 5 (N --> a r e) a new arc N is created for the phoneme sequence "a r e". From the initial state 0 with the nonterminal N, state 4 is determined from the goto table for the newly created arc. The action table indicates that the new state number is 4 and is expecting /o/, and is also ready for the reduce action (reduce 2). Therefore a new arc NP is creatd for the nonterminal NP.

From state 0 with the nonterminal NP, state 1 is determined from the goto table, whose state is expecting /o/ and /k/ (Figure 2-8).

Now we go back to Figure 2-7 and see how /e/ is connected to the other arc P_5. The action table indicates that the newly created arc for P_6 has the state number 19 and is ready for the action "reduce 4". Thus, using the rule 4 (N --> m a m e) the reduce action is executed and the new arc N is created. The goto table indicates that the state number for the newly created arc N is 4. Exactly the same situation occured when P_6 was connected to P_4. That is, both "mame" and "are" are reduced to N. We introduce the local ambiguity packing technique here:

> When the reduce action is executed on the condition that there exists the same nonterminal symbol whose last constituent (a phoneme in our task) and whose starting arc is the same, the arc for the reduce action is merged to the previously created arc.

In our example, the nonterminal arc N' for P_6 connected to P_4 is not created, because N' and the previously created arc N have the same phoneme P_6 as the last constituent and both N' and N are connected to the initial state 0. Therefore N and NP arcs in Figure 2-8 represent the phoneme sequence "are" and "mame". In other words N and NP are locally ambiguous. This local ambiguity packing technique avoids the exponential growth of parse trees.

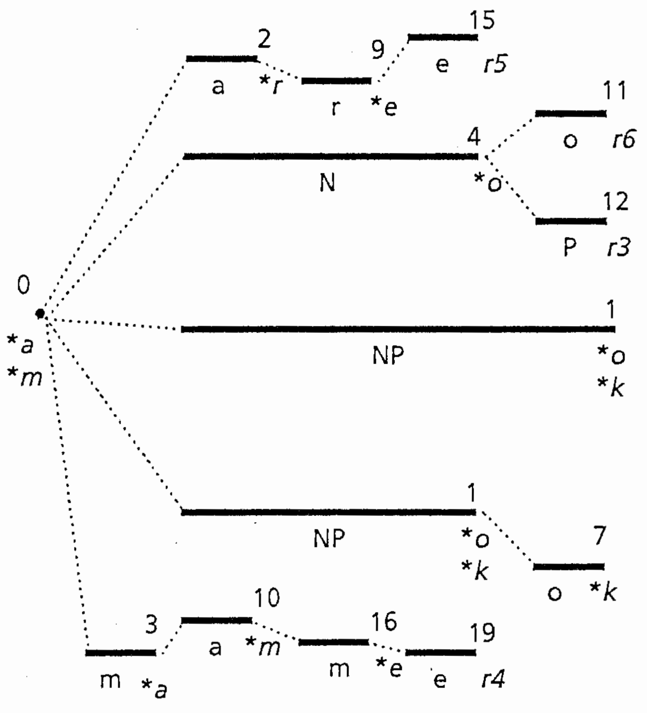

Figure 2-9

Now we continue parsing. The P_7 phoneme /o/ is expected by two arcs N and NP. Because the shifted states are different (ActionTable(4, o)=shift 11, ActionTable(1, o)=shift 7), two /o/ arcs are created for each, on the assumption that Junct(P_6, P_7) returns TRUE. The one /o/ arc connected to NP has the state number 7 and is expecting /k/ according to the action table. The other /o/ arc connected to N has the state number 11 and is ready for the action "reduce 6". Thus a new arc labeled P is created by the rule 6 (P --> o). The goto table gives the state number 12 for this arc. The action table indicates that state 10 is again ready for the action "reduce 3"(NP --> N P). Therefore a new arc labeled NP is created. Its state number is 1. State 1 is expecting /o/ and /k/ (Figure 2-9).

The parsing continues in this way. The final situation is shown in Figure 2-10. As

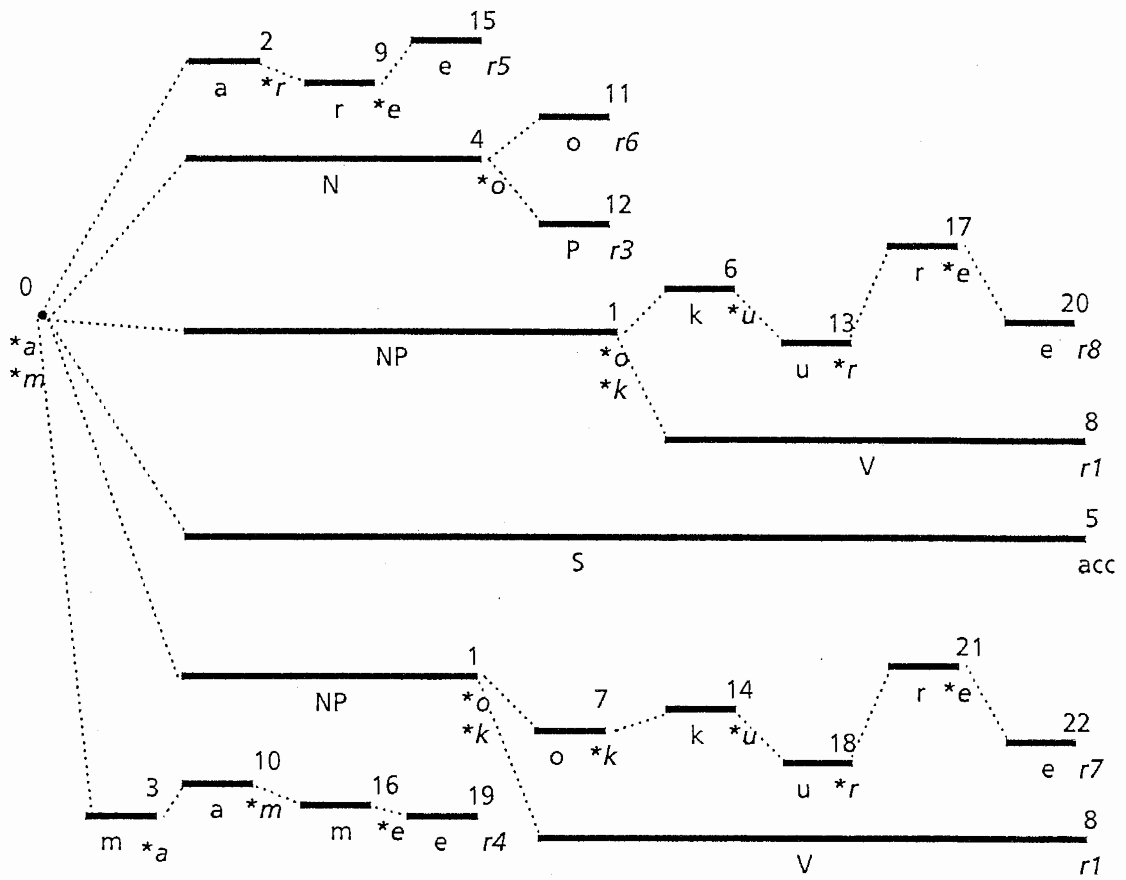

Figure 2-10

a result, the parser finds one packed arc S which contains the following four successful parses.

mame o kure
mame okure
are o kure
are okure

3. Experiment of Parsing a Phoneme Lattice

3.1. Lexicon and grammar

Since the terminal symbols of our grammar are phonemes, the grammar includes our domain's lexica. That is, a rule is represented as

```
    N --> s a N k a
```

instead of

```
    N --> saNka
```

whose form can be generally used for a context-free grammar of Natural Language Processing.

Current grammar consists of approximately 1500 rules and contains approximately 700 nouns, 150 verbs, 20 adjectives, 120 adverbs and 50 other words. Various conjugation forms of verbs and adjectives are also described in the grammar. For example, negative, past, passive forms, various forms represented by modal auxiliary verbs like may, must, can, shall, etc. are described.
The skeleton of our grammar, like the connection forms between verbs and auxiliary verbs, was written at the Center for Machine Translation at Carnegie Mellon University.

The following is a grammar excerpt which recognizes "ikura kakarimasuka"("How much does it cost?").

```
<S>                   -->  <ADV> <V>
<ADV>                 -->  i k u r a
<V>                   -->  <v-rentai-nonpast> k a
<v-rentai-nonpast>    -->  <v-rentai>
<v-rentai>            -->  <v-masu> s u
<v-masu>              -->  <v-renyo1> m a
<v-renyo1>            -->  <v-5dan-i>
<v-5dan-i>            -->  <v-5dan-r> r i
<v-5dan-r>            -->  k a k a                 ; verbal stem
```

3.2. A phoneme lattice

A phoneme lattice is obtained by the word spotting program developed by Kawabata [1].

The word spotting routine

```
    ws word -t threshold file
```

returns word candidates in the speech utterance file with the starting and ending frame number and with a score less than the threshold. Thus the output of this routine is the list of quadruplets such as:

((word$_1$ starting-frame$_1$ ending-frame$_1$ score$_1$)
 (word$_2$ starting-frame$_2$ ending-frame$_2$ score$_2$)

(word$_3$ starting-frame$_3$ ending-frame$_3$ score$_3$)
. . . . .)

Since this program is used for obtaining a phoneme lattice, the word papameter is a phoneme. The following phonemes are used:

/a/ /i/ /u/ /e/ /o/ /ei/ /ou/
/k/ /s/ /t/ /n/ /h/ /m/ /r/ /y/ /sh/ /chi/ /tsu/ /shi/
/ch/ /p/ /d/ /w/ /g/ /b/
/ky/ /hy/ /py/ /gy/ /zy/ /ny/ /ng/ /ngy/ /my/ /ry/ /N/ /Q/

3.3. Scoring

There are two main reasons why we want to score each parse: first, to prune the search space by discarding branches of the parse whose score is hopelessly low; second, to select the best sentence out of multiple candidates by comparing their scores.

As described in Section 2.3., the word spotting routine returns phonemes and scores. The more plausible phoneme has a lower score. We define the formula which calculates the score of partial parses as follows.

$$\sum_i (S_i - G) + Js + Je + \sum_i J_i \qquad (1)$$

Where
S_i is the score for each phoneme multiplied by 100
G is some integer(G can be considered as a kind of reward for each phoneme and G is set to 250 for our task.)
Js is the starting score
Je is the ending score
J_i is the juncture score for adjacent phonemes

The better partial parse has a lower score by the formula above. Although we assumed that Junct (PH_1, PH_2) returns only TRUE value in Section 2.3, the juncture validation function should also return some score along with TRUE. A validation function, however, is not avilable now. Thus, we temporarily define the function as the following:

Junct(PH_1, PH_2) returns
|PH_1-ending-frame-number - PH_2-starting-frame-number|
Junct(PH_1, PH_2) allows 20-frame overlap
Junct(PH_1, PH_2) allows 80-frame gap between phonemes
Junct(PH_1, PH_2) does not return TRUE when PH_2-starting-frame-number is smaller than PH_1-starting-frame-number
Junct(START, PH_2) returns TRUE when the PH_2-starting-frame-number is less than 55

Junct(PH_1, \$) returns TRUE when PH_1-ending-frame-number is greater than (end-of-speech - 70)

4. Experiment Result

The system has been tested against 279 phrases of the continuous speech data called "SB3". Two utterances of "ATR" are excluded from the data due to the shortage of training data for phoneme spotting module. A phoneme lattice was created for each phrase in advance, where each hypothesized phoneme has the score less than 3.00 except the /ng/ and /w/ phonemes(4.00 for these two). A phoneme lattice for each phrase contains 46 hypothesized phonemes in average.

4.1. Evaluation by the original pronunciation grammar

As described in Section 3.1, our context-free grammar is written on the phoneme base. First the grammar of the original pronunciation was used. Namely, a phrase "tourokusuru" is represented as "t ou r o k u s u r u." The result is shown in Table 4-1. Average parsing time was 3.0 seconds on a Symbolics Lisp Machine.

Recognition order	Recognition rate
First candidate	35 %
Within top 5 candidates	54 %
Within top 10 candidates	54 %
Within top 30 candidates	54 %

Table 4-1: Evaluation result 1

4.2. Handling erroneous phonemes

The evaluation above assumed that the phoneme is pronounced as it is written. The actual pronunciation, however, is not always the concatenation of individual phonemes. This situation is analyzed in the following:

First, an example of vowel-devocalization is considered. Two vowels /u/ and /i/ are devocalized in certain situations with a high probability[2]. For example, the first /u/ is often devocalized in the phrase "hukumarete." There are three ways to cope with this vowel-devocalization:

<Method 1> To invoke the word spotting program with the /hu/ phoneme.

<Method 2> To add the "h k u m a r e t e" sequence to the grammar in addition to the original pronunciation "h u k u m a r e t e."

<Method 3> To make the parser expect a /u/ phoneme when the /h/ phoneme is read.

Since each of the above ways has both merits and demerits, the right way should be chosen according to the situation:

<Situation A> When vowel-devocalization occurs with a very high probability independent of context, <Method 1> or <Method 2> is preferred. But using <Method 1> carelessly makes the phoneme spotting routine heavy and increases the size of the LR table. We adopted <Method 1> for the phonemes /shi/ and /tsu/ and <Method 2> for the phonemes /hu/, /su/ and /ku/.

<Situation B> When vowel-devocalization occurs dependent on the context with a high probability, <Method 2> is preferred.

<Situation C> When vowel-devocalization occurs occasionally, <Method 3> is preferred.

The categorization above is also valid for phoneme substitution/insertion/deletion other than vowel-devocalization. The meaning "to make the parser expect /u/ phoneme" in <Method 3> is described:

> Suppose that /h/ is read and that state expects /u/ according to the action table and the lattice does not contain /u/ immediately after /h/. On the condition above, a dummy arc /u/ connected to /h/ is created. A score greater than G in formula (1) is given to the dummy arc as a penalty.

4.3. Evaluation by grammar with phoneme modification

The evaluation result when the grammar allows /h/, /k/, /s/, and /ng/ substitution for /hu/, /ku/, /su/, and /Nng/ respectively is shown in Table 4-2. Average parsing time was 3.1 seconds.

Recognition order	Recognition rate
First candidate	45 %
Within top 5 candidates	65 %
Within top 10 candidates	66 %
Within top 30 candidates	66 %

Table 4-2: Evaluation result 2

4.4. Evaluation with various phoneme deletions/substitutions considered

The evaluation result on the following conditions is shown in Table 4-3. Average

Recognition order	Recognition rate
First candidate	51 %
Within top 5 candidates	82 %
Within top 10 candidates	85 %
Within top 30 candidates	87 %

Table 4-3: Evaluation result 3

parsing time was 12.9 seconds.

(1) The grammar allows /h/, /k/, /s/, and /ng/ substitution for /hu/, /ku/, /su/, and /Nng/ respectively(the same condition in Section 4.3).

(2) The grammar also allows /ny/, /my/, and /zy/ substitution for /ni/, /mi/, and /zi/ respectively.

(3) The parser is made to assume that the /o/ and /u/ phonemes might be missing after a /y/ phoneme with a high probability.

(4) The parser is made to assume that the /a/, /i/, /e/, /o/ and /u/ phonemes might be missing after a /r/ phoneme with a high probability.

(5)The parser is made to assume that the /a/, /i/, /e/ and /o/ phonemes might be missing after a /k/ phoneme with a high probability.

(6)The parser is made to assume that the /i/ phoneme might be missing after a /h/ phoneme with a high probability.

4.5. Unrecognized examples

A few examples where the parser cannot find the right phrase are presented in the following.

Phrase 22: "omoushikomi"
The very first /o/ is missing in the lattice. Thus "moushikomi" is found.

Phrase 75: "choukounomino"
Only the /o/ phoneme between /n/ and /m/ is missing in the lattice. The method for handling this situation is presented in Section 5.

Phrase 80: "yokoushuudaio"
The first /o/ phoneme is missing in the lattice. Even when the dummy arc /o/ is created(as in Section 4.4), that partial parse is pruned because of the low score. Less strict pruning is desired.

5. Concluding Remarks

An experiment in producing phrase candidates from a phoneme lattice is presented. There are some ways to improve the current system:

(1) Because the word spotting routine used was not developed for phoneme spotting, specialized phoneme spotting system is desired.

(2) A phoneme lattice is created in advance this time. But pipe-line processing can be performed when the parser is connected to the real-time phoneme spotting module, because the adopted parsing method reads the hypothesized phonemes from the starting point one by one.

(3) The current parser uses the primitive juncture validation function which only examines the gap and overlap by frame number. A predicate which analyzes the speech signal will surely improve the accuracy.

(4) An LR table can predict the next phoneme(s) at any state. If a lattice does not contain the phoneme which a hopeful partial parse expects, the parser can insert that phoneme. The problem of arbitrary missing phonemes can be solved by this method.

(5) The current grammar only handles the syntax. A semantic check and a more accurate syntax check will prune inappropriate parses. A semantic handler can easily be embeded in a context-free grammar and that semantic check will aid a higher level of Natural Language Processing like semantic extraction [3].

Acknowledgments

The author wishes to thank Mr. Kenji Kita and Dr. Takeshi Kawabata for the phoneme spotting program. The author would also like to express gratitude to Dr. Akira Kurematsu, president of ATR Interpreting Telephony Research Laboratories and to Dr. Kiyohiro Shikano, head of the Speech Processing Department for their encouragement and support, which made this research possible.

References

[1] Kawabata, T., Hanazawa, T. and Shikano, K.
Word Spotting Method Based on HMM Phoneme Spotting.
Paper of Technical Group, SP88-23, IEICE. June, 1988(in Japanese).

[2] Kuwabara, H. and Takeda, K.
Analysis and Prediction of Vowel Devocalization in Isolated Japanese Words.
ATR Technical Report TR-I-0030. June, 1988.

[3] Saito, H. and Tomita, M.
Parsing Noisy Sentences.
12th International Conference on Computational Linguistics (COLING 88). Budapest, August, 1988.

[4] Tomita, M.
Efficient Parsing for Natural Language: A Fast Algorithm for Practical Systems.
Kluwer Academic Publishers. Boston, MA, 1985.

[5] Tomita, M.
An Efficient Word Lattice Parsing Algorithm for Continuous Speech Recognition.
IEEE-IECEJ-ASJ International Conference on Acoustics, Speech, and Signal Processing (ICASSP86). Tokyo, April, 1986.

Appendix. Sample Runs

Two actual outputs of the parser are shown on the following pages. The right phrases are "ikura" for DATA 64 and "kakarimasuka" for DATA 65.

```
DATA 64: 22 phonemes in the lattice.

--- Initialization ---

--- Start building words ---

Evaluation of (PROCESSWORD) took 7.978648 seconds of elapsed time
including 0.938 seconds waiting for the disk for 0 faults.
The garbage collector has flipped, so consing was not measured.
--- End of Parsing ---

 130 words found

1: (-526.0) I<20-35 #1.41> K<43-72 #1.57> U<73-85 #1.27> R<85-90 #1.54> A<99-120 #1.07>
2: (-394.0) H<41-55 #1.87> K<43-72 #1.57> U<73-85 #1.27> D<85-102 #2.21> A<99-120 #1.07>
3: (-361.0) K<1-18 #2.6> I<20-35 #1.41> K<43-72 #1.57> (A<72-82>) R<85-90 #1.54> A<99-120 #1.07>
4: (-341.0) K<43-72 #1.57> U<73-85 #1.27> D<85-102 #2.21> A<99-120 #1.07>
5: (-313.0) K<1-18 #2.6> (A<18-28>) I<20-35 #1.41> K<43-72 #1.57> (A<72-82>) R<85-90 #1.54> A<99-
120 #1.07>
6: (-303.0) K<1-18 #2.6> I<20-35 #1.41> K<43-72 #1.57> U<73-85 #1.27>
6: (-303.0) R<29-34 #2.52> (O<34-44>) K<43-72 #1.57> U<73-85 #1.27> D<85-102 #2.21> A<99-120 #1.0
7>
6: (-303.0) R<29-34 #2.52> (O<34-44>) K<43-72 #1.57> U<73-85 #1.27> D<85-102 #2.21> A<99-120 #1.0
7>
9: (-296.0) I<20-35 #1.41> K<43-72 #1.57> U<73-85 #1.27>
10: (-293.0) I<20-35 #1.41> K<43-72 #1.57> (A<72-82>) D<85-102 #2.21> A<99-120 #1.07>
10: (-293.0) I<20-35 #1.41> K<43-72 #1.57> (I<72-82>) D<85-102 #2.21> A<99-120 #1.07>
12: (-288.0) H<41-55 #1.87> R<68-73 #2.63> U<73-85 #1.27> D<85-102 #2.21> A<99-120 #1.07>
13: (-271.0) K<1-18 #2.6> (I<18-28>) R<29-34 #2.52> (O<34-44>) K<43-72 #1.57> U<73-85 #1.27> D<85
-102 #2.21> A<99-120 #1.07>
14: (-251.0) H<41-55 #1.87> (I<55-65>) R<68-73 #2.63> U<73-85 #1.27> D<85-102 #2.21> A<99-120 #1.
07>
15: (-226.0) K<1-18 #2.6> U<73-85 #1.27> D<85-102 #2.21> A<99-120 #1.07>
16: (-225.0) H<41-55 #1.87> K<43-72 #1.57> U<73-85 #1.27>
17: (-218.0) K<1-18 #2.6> (O<18-28>) R<68-73 #2.63> U<73-85 #1.27> D<85-102 #2.21> A<99-120 #1.07
>
17: (-218.0) K<43-72 #1.57> U<73-85 #1.27> R<85-90 #1.54> (U<90-100>)
17: (-218.0) K<43-72 #1.57> U<73-85 #1.27> R<85-90 #1.54> (E<90-100>)
20: (-212.0) K<1-18 #2.6> (I<18-28>) K<43-72 #1.57> (A<72-82>) R<85-90 #1.54> A<99-120 #1.07>
21: (-190.0) H<41-55 #1.87> K<43-72 #1.57> U<73-85 #1.27> E<77-99 #2.77>
21: (-190.0) K<1-18 #2.6> (O<18-28>) R<29-34 #2.52> U<73-85 #1.27> D<85-102 #2.21> A<99-120 #1.07
>
23: (-181.0) K<1-18 #2.6> I<20-35 #1.41> K<43-72 #1.57>
24: (-177.0) K<1-18 #2.6> I<20-35 #1.41> K<43-72 #1.57> (E<72-82>) R<85-90 #1.54> (U<90-100>)
24: (-177.0) K<1-18 #2.6> I<20-35 #1.41> K<43-72 #1.57> (E<72-82>) R<85-90 #1.54> (U<90-100>)
24: (-177.0) K<1-18 #2.6> I<20-35 #1.41> K<43-72 #1.57> (E<72-82>) R<85-90 #1.54> (E<90-100>)
24: (-177.0) K<1-18 #2.6> I<20-35 #1.41> K<43-72 #1.57> (E<72-82>) R<85-90 #1.54> (E<90-100>)
28: (-174.0) I<20-35 #1.41> K<43-72 #1.57>
29: (-173.0) K<1-18 #2.6> I<20-35 #1.41> R<68-73 #2.63> U<73-85 #1.27>
29: (-173.0) H<41-55 #1.87> U<73-85 #1.27> R<85-90 #1.54> (I<90-100>)

DATA 65: 51 phonemes in the lattice.

--- Initialization ---

--- Start building words ---

Evaluation of (PROCESSWORD) took 36.945295 seconds of elapsed time
including 4.661 seconds waiting for the disk for 0 faults.
The garbage collector has flipped, so consing was not measured.
--- End of Parsing ---

 1591 words found

1: (-799.0) K<12-29 #2.29> (A<29-39>) K<43-65 #1.58> A<62-83 #1.31> R<81-86 #1.37> I<94-109 #1.64
> M<119-137 #1.79> A<133-154 #1.
59> S<165-190 #0.72> K<190-209 #2.5> A<211-232 #1.2>
2: (-781.0) K<12-29 #2.29> (A<29-39>) K<43-65 #1.58> A<62-83 #1.31> R<81-86 #1.37> I<94-109 #1.64
> M<119-137 #1.79> A<133-154 #1.
59> S<165-190 #0.72> U<178-190 #2.56> K<190-209 #2.5> A<211-232 #1.2>
3: (-776.0) K<12-29 #2.29> (I<29-39>) K<43-65 #1.58> A<62-83 #1.31> R<81-86 #1.37> E<81-103 #1.84
> M<119-137 #1.79> A<133-154 #1.
59> S<165-190 #0.72> K<190-209 #2.5> A<211-232 #1.2>
3: (-776.0) K<12-29 #2.29> (A<29-39>) K<43-65 #1.58> A<62-83 #1.31> R<81-86 #1.37> E<81-103 #1.84
> M<119-137 #1.79> A<133-154 #1.
59> S<165-190 #0.72> K<190-209 #2.5> A<211-232 #1.2>
```

```
3: (-776.0) K<12-29 #2.29> (A<29-39>) K<43-65 #1.58> A<62-83 #1.31> R<81-86 #1.37> E<81-103 #1.84
> M<119-137 #1.79> A<133-154 #1.
59> S<165-190 #0.72> K<190-209 #2.5> A<211-232 #1.2>
6: (-762.0) K<1-18 #2.77> (A<18-28>) K<43-65 #1.58> A<62-83 #1.31> R<81-86 #1.37> I<94-109 #1.64>
 M<119-137 #1.79> A<133-154 #1.5
9> S<165-190 #0.72> K<190-209 #2.5> A<211-232 #1.2>
7: (-758.0) K<12-29 #2.29> (I<29-39>) K<43-65 #1.58> A<62-83 #1.31> R<81-86 #1.37> E<81-103 #1.84
> M<119-137 #1.79> A<133-154 #1.
59> S<165-190 #0.72> U<178-190 #2.56> K<190-209 #2.5> A<211-232 #1.2>
7: (-758.0) K<12-29 #2.29> (A<29-39>) K<43-65 #1.58> A<62-83 #1.31> R<81-86 #1.37> E<81-103 #1.84
> M<119-137 #1.79> A<133-154 #1.
59> S<165-190 #0.72> U<178-190 #2.56> K<190-209 #2.5> A<211-232 #1.2>
7: (-758.0) K<12-29 #2.29> (A<29-39>) K<43-65 #1.58> A<62-83 #1.31> R<81-86 #1.37> E<81-103 #1.84
> M<119-137 #1.79> A<133-154 #1.
59> S<165-190 #0.72> U<178-190 #2.56> K<190-209 #2.5> A<211-232 #1.2>
10: (-744.0) K<1-18 #2.77> (A<18-28>) K<43-65 #1.58> A<62-83 #1.31> R<81-86 #1.37> I<94-109 #1.64
> M<119-137 #1.79> A<133-154 #1.
59> S<165-190 #0.72> U<178-190 #2.56> K<190-209 #2.5> A<211-232 #1.2>
11: (-739.0) K<1-18 #2.77> (I<18-28>) K<43-65 #1.58> A<62-83 #1.31> R<81-86 #1.37> E<81-103 #1.84
> M<119-137 #1.79> A<133-154 #1.
59> S<165-190 #0.72> K<190-209 #2.5> A<211-232 #1.2>
11: (-739.0) K<1-18 #2.77> (A<18-28>) K<43-65 #1.58> A<62-83 #1.31> R<81-86 #1.37> E<81-103 #1.84
> M<119-137 #1.79> A<133-154 #1.
59> S<165-190 #0.72> K<190-209 #2.5> A<211-232 #1.2>
11: (-739.0) K<1-18 #2.77> (A<18-28>) K<43-65 #1.58> A<62-83 #1.31> R<81-86 #1.37> E<81-103 #1.84
> M<119-137 #1.79> A<133-154 #1.
59> S<165-190 #0.72> K<190-209 #2.5> A<211-232 #1.2>
14: (-736.0) K<5-22 #2.99> (A<22-32>) K<43-65 #1.58> A<62-83 #1.31> R<81-86 #1.37> I<94-109 #1.64
> M<119-137 #1.79> A<133-154 #1.
59> S<165-190 #0.72> K<190-209 #2.5> A<211-232 #1.2>
15: (-724.0) H<16-30 #2.11> (I<30-40>) R<36-41 #2.66> (A<41-51>) K<43-65 #1.58> A<62-83 #1.31> R<
81-86 #1.37> E<81-103 #1.84> M<1
19-137 #1.79> A<133-154 #1.59> S<165-190 #0.72> K<190-209 #2.5> A<211-232 #1.2>
16: (-721.0) K<1-18 #2.77> (I<18-28>) K<43-65 #1.58> A<62-83 #1.31> R<81-86 #1.37> E<81-103 #1.84
> M<119-137 #1.79> A<133-154 #1.
59> S<165-190 #0.72> U<178-190 #2.56> K<190-209 #2.5> A<211-232 #1.2>
16: (-721.0) K<1-18 #2.77> (A<18-28>) K<43-65 #1.58> A<62-83 #1.31> R<81-86 #1.37> E<81-103 #1.84
> M<119-137 #1.79> A<133-154 #1.
59> S<165-190 #0.72> U<178-190 #2.56> K<190-209 #2.5> A<211-232 #1.2>
16: (-721.0) K<1-18 #2.77> (A<18-28>) K<43-65 #1.58> A<62-83 #1.31> R<81-86 #1.37> E<81-103 #1.84
> M<119-137 #1.79> A<133-154 #1.
59> S<165-190 #0.72> U<178-190 #2.56> K<190-209 #2.5> A<211-232 #1.2>
19: (-718.0) K<5-22 #2.99> (A<22-32>) K<43-65 #1.58> A<62-83 #1.31> R<81-86 #1.37> I<94-109 #1.64
> M<119-137 #1.79> A<133-154 #1.
59> S<165-190 #0.72> U<178-190 #2.56> K<190-209 #2.5> A<211-232 #1.2>
20: (-713.0) K<5-22 #2.99> (I<22-32>) K<43-65 #1.58> A<62-83 #1.31> R<81-86 #1.37> E<81-103 #1.84
> M<119-137 #1.79> A<133-154 #1.
59> S<165-190 #0.72> K<190-209 #2.5> A<211-232 #1.2>
20: (-713.0) K<5-22 #2.99> (A<22-32>) K<43-65 #1.58> A<62-83 #1.31> R<81-86 #1.37> E<81-103 #1.84
> M<119-137 #1.79> A<133-154 #1.
59> S<165-190 #0.72> K<190-209 #2.5> A<211-232 #1.2>
20: (-713.0) K<5-22 #2.99> (A<22-32>) K<43-65 #1.58> A<62-83 #1.31> R<81-86 #1.37> E<81-103 #1.84
> M<119-137 #1.79> A<133-154 #1.
59> S<165-190 #0.72> K<190-209 #2.5> A<211-232 #1.2>
23: (-706.0) H<16-30 #2.11> (I<30-40>) R<36-41 #2.66> (A<41-51>) K<43-65 #1.58> A<62-83 #1.31> R<
81-86 #1.37> E<81-103 #1.84> M<1
19-137 #1.79> A<133-154 #1.59> S<165-190 #0.72> U<178-190 #2.56> K<190-209 #2.5> A<211-232 #1.2>
24: (-695.0) K<5-22 #2.99> (I<22-32>) K<43-65 #1.58> A<62-83 #1.31> R<81-86 #1.37> E<81-103 #1.84
> M<119-137 #1.79> A<133-154 #1.
59> S<165-190 #0.72> U<178-190 #2.56> K<190-209 #2.5> A<211-232 #1.2>
24: (-695.0) K<5-22 #2.99> (A<22-32>) K<43-65 #1.58> A<62-83 #1.31> R<81-86 #1.37> E<81-103 #1.84
> M<119-137 #1.79> A<133-154 #1.
59> S<165-190 #0.72> U<178-190 #2.56> K<190-209 #2.5> A<211-232 #1.2>
24: (-695.0) K<5-22 #2.99> (A<22-32>) K<43-65 #1.58> A<62-83 #1.31> R<81-86 #1.37> E<81-103 #1.84
> M<119-137 #1.79> A<133-154 #1.
59> S<165-190 #0.72> U<178-190 #2.56> K<190-209 #2.5> A<211-232 #1.2>
27: (-671.0) K<12-29 #2.29> (A<29-39>) K<43-65 #1.58> A<62-83 #1.31> R<81-86 #1.37> I<94-109 #1.6
4> M<119-137 #1.79> A<133-154 #1
.59> S<165-190 #0.72>
28: (-653.0) K<12-29 #2.29> (A<29-39>) K<43-65 #1.58> A<62-83 #1.31> R<81-86 #1.37> I<94-109 #1.6
4> M<119-137 #1.79> A<133-154 #1
.59> S<165-190 #0.72> U<178-190 #2.56>
29: (-650.0) K<12-29 #2.29> (I<29-39>) R<36-41 #2.66> A<62-83 #1.31> R<81-86 #1.37> E<81-103 #1.8
4> M<119-137 #1.79> A<133-154 #1
.59> S<165-190 #0.72> K<190-209 #2.5> A<211-232 #1.2>
29: (-650.0) K<12-29 #2.29> (O<29-39>) R<36-41 #2.66> A<62-83 #1.31> R<81-86 #1.37> E<81-103 #1.8
```