人工知能学会研究会資料
SIG-FAI-9401-3(6/6)

Incorporation of Phoneme-Context-Dependence in LR Table through Constraint Propagation Method

TANAKA Hozumi, LI Hui and TOKUNAGA Takenobu

Department of Computer Science
Tokyo Institute of Technology
2-12-1 Ôokayama Meguro Tokyo 152 Japan

Abstract

It is obvious that successful speech recognition requires the use of linguistic information. For this purpose, a generalized LR (GLR) parser provides an exceptionally competent and flexible framework to combine linguistic information with phonological information.

The combination of a GLR parser and allophone models is considered very effective for enhancing the recognition accuracy in a large vocabulary continuous speech recognition. The main problem of integrating GLR parsing into an allophone-based recognition system is how to solve the word juncture problem, that is, how to express the phones at a word boundary with allophone models.

This paper proposes a new method called CPM (Constraint Propagation Method) to generate an allophone-based LR table, which can effectively solve the word juncture problem. In our method, by introducing the allophone rules into the CFG and lexical rules, an LR table is generated, then the LR table is modified on the basis of an allophone connection matrix by applying the constraint propagation method. With this modified LR table, precise allophone predictions for speech recognition can be obtained.

1 Introduction

It is obvious that successful speech recognition requires the use of linguistic information. For this purpose, a generalized LR (GLR) parser provides an exceptionally competent and flexible framework to combine linguistic information with phonological information.

One difficulty with large vocabulary continuous speech recognition is to reduce the search space. The GLR parser can meet this requirement[4][3] by applying linguistic constraints to speech recognition. In the phone-based speech recognition system, the GLR parser has been employed as a phoneme predictor, which provides efficient search of phones in the process of speech recognition. The GLR parser is guided by an LR table automatically generated from context-free grammar (CFG) rules and proceeds left-to-right without backtracking. In order to make phone predictions, the lookahead symbols in the table are phones instead of the usual grammatical categories. Thus, lexical rules are expressed as follows:

$<$*grammatical category*$>$ $\rightarrow$ $<$*a sequence of phones*$>$.

Several experiments[3][6][7][8] have shown that the performance of speech recognition systems could be improved by using allophones as recognition units instead of phones. Allophone models (such as triphone models) are context-dependent phone models that take into consideration the left and the right neighboring phones to model the major coarticulatory effects in continuous speech.

The combination of allophone models and a GLR parser is desirable to achieve better performance in continuous speech recognition. The main problem of integrating GLR parsing into allophone-based recognition system is how to solve the word juncture problem, that is, how to express the phones at a word boundary with allophone models.

In this paper, we propose a new method to generate an allophone-based LR table that can solve the word juncture problem. This method, by introducing a set of allophone rules into the CFG and lexical rules, generates an LR table, then modifies the LR table on the basis of an allophone connection matrix by applying the constraint propagation method (CPM).

The organization of this paper is as follows: Section 2 provides an overview of the past allophone-based GLR parsing methods and points out problems in those methods; after discussing the advantages of using the canonical LR table for speech recognition, Section 3 describes our method to generate an allophone-based

LR table by applying CPM; Section 4 provides comparisons between the LR tables before and after CPM; and Section 5 concludes with future works.

In the following sections, we will use several examples from Japanese, but the method we propose is not language specific – it can be applied to many spoken languages.

2 Overview of Allophone-Based GLR Parsing Method

The main problem of integrating GLR parsing into an allophone-based recognition system is solving the word juncture problem. Consider the Japanese word "a k i", which is a sequence of three phones in the lexical rule that follows:

$$\textit{noun} \rightarrow a\ k\ i. \quad \textit{(autumn)}$$

The allophone of "k", within the word, can be determined by the left and right context, which are known in advance, namely "a" and "i" respectively. On the other hand, the phones "a" and "i" are located at the word boundary. For the beginning phone "a", there is no left context and for the end phone "i", there is no right context and, thus, with phones at word boundaries, it is difficult to know the left or right context beforehand. To solve this problem, several allophone-based GLR parsing methods have been proposed[3][7].

Itou et al.[3] have used the lexical rules that follow:

$$\textit{noun} \rightarrow a(*,k)\ k1\ i(k,*)$$

where k1 is an allophone of "k", which has the left and the right context "a" and "i"; a(*,k) is a special phone of "a" whose right context is known, and i(k,*) is a special phone of "i" whose left context is known. The LR table is constructed from a CFG and lexical rule set. The allophones at word boundaries are determined dynamically in the recognition process when the succeeding and preceding words are obtained.

Some changes in a GLR parsing algorithm are required to take account of the phoneme-context-dependence at word boundaries. Furthermore, for the end phones i(k,*) in the above example, the system makes needless allophone predictions because of no right context.

Nagai et al.[7] have proposed the three approaches that follow :

(1)Grammar level realization

As well as Itou's method, phones within a word are changed into allophones. Phones at a word boundary are changed into possible allophones, which generate new grammatical categories. For instance, if "a(*,k)" and "i(k,*)" both have two allophones, "a1" and "a2", "i1" and "i2" respectively, then the four lexical rules are created from a word such as "a k i" are as follows:

$$\textit{a1\_noun\_i1} \rightarrow \textit{a1 k1 i1}, \quad \textit{a2\_noun\_i1} \rightarrow \textit{a2 k1 i1}$$
$$\textit{a1\_noun\_i2} \rightarrow \textit{a1 k1 i2}, \quad \textit{a2\_noun\_i2} \rightarrow \textit{a2 k1 i2}$$

Therefore, too many lexical rules are created from each word. The creation of new grammatical categories will produce many new CFG rules. Furthermore, nonterminal symbols of RHS (right hand side) in the CFG rule must take account of the word juncture problem considering newly created nonterminal symbols. For example, the nonterminal symbols, "i_cat_j" and "j_cat'_k" can be adjacent in this order, and so on. Thus, phoneme-context-dependence is expressed by a large number of lexical and grammar rules. Although this method requires no change of a GLR parsing algorithm, the explosion of states in an LR table may occur.

(2)Table level realization

From a set of CFG and phoneme-context-independent lexical rules, this method generates an LR table, then it introduces a set of allophones step by step by adding new states and new nonterminal symbols in the LR table to incorporate phoneme-context-dependence. The table modification process is complex and requires changes to the parsing algorithm to keep the left and right context of allophones. Thus, phoneme-context-dependence must be dynamically recognized in the parsing process. This method results in the increase of the number of states in the LR table and brings inefficiency to the parsing process.

(3) Parsing level realization

In this method, the phoneme-context-dependence is not incorporated in an LR table. The recognition of allophones is completely handled by a GLR parser, which has to include a method of the phoneme-context-dependence in a procedural way. This makes the original GLR parsing algorithm more complex and inefficient.

3 Allophone-Based LR Table after Applying Constraint Propagation Method

In this section, we propose a new method called CPM (Constraint Propagation Method) to generate an allophone-based LR table that can solve the word juncture problem and enables precise allophone predictions. This method consists of the four steps:

(1) Construction of an allophone connection matrix that provides the connectability between adjacent allophones.

(2) Phones within a word in lexical rules are changed into allophones.

(3) In addition to the above lexical and CFG rules, allophone rules are introduced to generate an allophone-based LR table.

(4) The allophone-based LR table is modified with the allophone connection matrix applying CPM, and, finally, the modified LR table is compressed to reduce the table size.

Before explaining the details of each step, we would like to discuss the types of LR tables. All methods mentioned in Section 2 have used the SLR or LALR table. Compared with the canonical LR table, the SLR and LALR table can not provides the precise phoneme predictions because the SLR and LALR table have fewer states due to merging several states in an LR table [1], and merging several states brings many actions in a state that produces many predictions.

This is why our method uses the canonical LR table. Generally, the canonical LR table has more states than the SLR or LALR table, but by applying Step 4, we achieve reduction of the table size.

Step 1: Creation of the allophone connection matrix

The allophone context of an allophone "x" is defined as:

<left context> *x* *<right context>*

where *<left context>* and *<right context>* are a set of phones. We assume that there is only one allophone context for one allophone. An allophone connection matrix is created from a set of allophone contexts.

Let us consider allophone "i2" of phone "i", allophone "d1" of phone "d", and the allophone contexts shown below.

$$\left\{\begin{matrix}\vdots\\ ch\\ \vdots\end{matrix}\right\} \mathrm{i2} \left\{\begin{matrix}\vdots\\ d\\ \vdots\end{matrix}\right\}, \quad \left\{\begin{matrix}\vdots\\ i\\ \vdots\end{matrix}\right\} \mathrm{d1} \left\{\begin{matrix}a\\ \vdots\\ \vdots\end{matrix}\right\}$$

According to the above allophone contexts in the form of triphone, "d1" can follow "i2" because the right context of "i2" contains the phone "d" and the left context of "d1" contains the phone "i".

If a connection matrix is expressed as an array of *Connect*[left_allophone, right_allophone], we can fill *Connect*[i2,d1] with symbol "1" to indicate that "i2" and "d1" are connectable in this order. Therefore, we can construct an allophone connection matrix from a set of allophone contexts. Note: an entry in the matrix, which can not be filled with "1", is marked by "0" to indicate the nonconnectability between the corresponding two adjacent allophones.

		R I G H T										
		h1	h2	a1	a2	ch1	ch2	i1	i2	d1	d2	$
L E F T	h1			1	0							
	h2			0	0							
	a1	1	1							1	1	1
	a2	0	0							0	0	0
	ch1							0	0			
	ch2							0	1			
	i1					0	0			1	0	
	i2					1	1			1	0	
	d1			1	0							
	d2			0	0							

Fig. 1 An example of the allophone connection matrix partially filled

Fig. 1 is an example of the allophone connection matrix partially filled. We will use this connection matrix to incorporate allophone connection constraints into the LR table at Step 4 by applying CPM.

Step 2: Conversion of the lexical rules

By using an example of simple Japanese CFG and lexical rules shown in Fig. 2, we can illustrate the conversion of the lexical rules. The lexical rules are from (2) to (4).

```
(1)  S → N BE          (3)  N → ch i ch i
                                 (father)
(2)  N → h a h a       (4)  BE → d a
        (mother)                 (be)
```

Fig. 2 An example of the CFG rules and lexical rules

The phones within a word are automatically converted into the allophones using a set of allophone contexts. We can not, however, change the phones at word boundaries because of the word juncture problem. Therefore, the lexical rules in Fig. 2, become those shown in Fig. 3 after Step 2.

```
(2)'  N → h a1 h1 a      (4)'  BE → d a
(3)'  N → ch i2 ch2 i
```

Fig. 3 Conversion of the lexical rules

	ACTION											GOTO							
state	h1	h2	a1	a2	ch1	ch2	i1	i2	d1	d2	$	h	a	ch	i	d	N	BE	S
0	sh5	sh6(c)			sh2(c)	sh3						4		1			7		25
1								sh8											
2								re9(a)											
3								re10											
4		(c)	sh9																
5			re5																
6			re6(a)																
7									sh11	sh12(c)						10		13	
8						sh21													
9	sh17																		
10			sh15	sh16(c)									14						
11			re13	re13(a)															
12			re14(a)	re14(a)						(d)									
13											re1								
14						(b)					re4								
15											re7								
16											re8(a)								
17			sh19	sh20(b)									18						
18									re2	re2(d)									
19									re7	re7(d)	(d)								
20									re8(a)	re8(a)									
21							sh23(b)	sh24							22				
22									re3	re3(d)									
23							(f)		re11(f)	re11(a)									
24									re12	re12(a)									
25											acc								

Fig. 4 Canonical LR(CLR) table generated from rules in Fig. 6

Step 3: Generation of the allophone-based LR table

The allophone rules are derived by pairing a phone with the corresponding allophones:

$<phone> \rightarrow <allophone1> \mid <allophone2> \mid \cdots$

Assume the following: {h1, h2} for "h", {a1, a2} for "a", {ch1, ch2} for "ch", {i1, i2} for "i", {d1, d2} for "d", a set of allophone rules from (5) to (14) in Fig. 5 can be produced.

(5) h → h1	(10) ch → ch2
(6) h → h2	(11) i → i1
(7) a → a1	(12) i → i2
(8) a → a2	(13) d → d1
(9) ch → ch1	(14) d → d2

Fig. 5 A set of the allophone rules

Now we have a set of CFG rules, lexical rules, and allophone rules as shown in Fig. 6

(1) S → N BE	(8) a → a2
(2)' N → h a1 h1 a	(9) ch → ch1
(3)' N → ch i2 ch2 i	(10) ch → ch2
(4)' BE → d a	(11) i → i1
(5) h → h1	(12) i → i2
(6) h → h2	(13) d → d1
(7) a → a1	(14) d → d2

Fig. 6 A set of the CFG, lexical and allophone rules

From Fig. 6, we generate a canonical LR table (see Fig. 4). Note: all lookahead symbols in this table are allophones.

Step 4: Constraint Propagation

The canonical LR table in Fig. 4 does not include any allophone connection constraints as shown in Fig. 1.

In order to incorporate these connection constraints into the LR table, we combine our CPM with the method developed by Tanaka et al.[9] and modify the original canonical LR table. Initial constraints are imposed on the LR table by the allophone connection matrix, then these constraints propagate throughout the table to produce a modified allophone-based LR table.

The outline of our method is shown in Fig. 7.

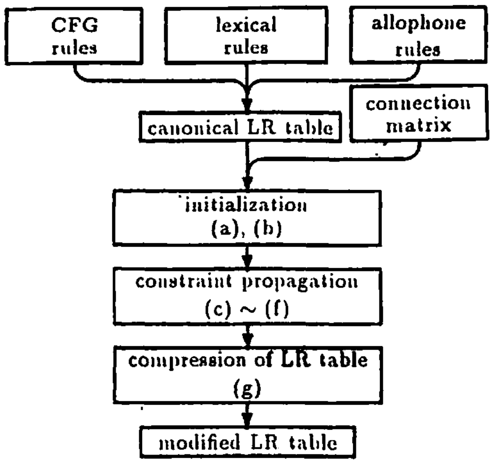

Fig. 7 Outline of CPM

(a). Deletable reduce actions with allophone rules

The constraint propagation starts checking every reduce action with allophone rules by employing the connection matrix in Fig. 1. According to the method developed by Tanaka et al.[9], the illegal reduce actions, which violate connection constraints, are marked "deletable". Fig. 8 shows this procedure.

```
for each reduce action R with an allophone rule
  in ecah entry of LR table {
    if (Connect[x, y] = 0) {
       mark R "deletable";
    }
}
where
  x: the RHS of the allophone rule used by R.
  y: the lookahead symbol of R.
  Connect: the allophone connection matrix.
```

Fig. 8 Checking the reduce actions with allophone rules

Consider, for example, re6 in state 6 and column "a1" in Fig. 4. The allophone connection matrix indicates that the connection between "h2" (RHS of rule 6) and "a1" is not allowed(*Connect*[h2,a1] = 0), so this reduce action re6 is illegal, and is marked "deletable".

In Fig. 4, all the deletable reduce actions found by this substep are marked (a).

(b). Deletable shift actions whose predecessors are shift actions

Let us consider the left of the end phone of a word. If the left phone is an allophone, we can easily check the connectability between the left allophone and its succeeding end phone. In this case, after shifting the left allophone, we have to immediately shift all the possible allophones belonging to the end phone. Consecutive shift actions will occur.

Consider a Japanese word "ch i2 ch2 i", and assume that there are two possible allophones "i1" and "i2" for the end phone "i". The left allophone of the end phone "i" is "ch2". After shifting "ch2" by sh21 in state 8, we have to shift "i1" and "i2". In Fig. 4, sh23 in state 21 is to shift "i1". *Connect*[ch2, i1]=0 means, however, that shifting "i1" is not allowed. Therefore, sh23 in state 21 is not allowed and is marked "deletable". On the contrary, sh24 with lookahead symbol "i2" in state 21 is not "deletable", because *Connect*[ch2, i2]=1. Fig. 9 shows this procedure.

```
for each shift action S in ecah entry of LR table {
   if (the action prior to S is a shift action) {
      if (Connect[x, y] = 0) {
         mark S "deletable";
      }
   }
}
where
 x: the lookahead symbol of the shift action prior to S.
 y: the lookahead symbol of S.
 Connect: the allophone connection matrix.
```

Fig. 9 Checking the shift actions whose predecessors are shift actions

In Fig. 4, all the deletable shift actions found by this substep are marked (b).

(c). Deletable shift actions that lead to empty states

An empty state is a state whose actions are all marked "deletable". After the above two substeps, (a) and (b), if there is an empty state the shift actions that lead to this empty state should be marked "deletable".

For example, in Fig. 4, sh6 in state 0 with the lookahead symbol "h2" should be marked "deletable", since state 6 is an empty state. All the deletable actions found by this substep are marked (c).

(d). Deletable reduce actions that lead to deletable shift actions

If a sequence of the reduce actions reaches a specific shift action, each reduce action in the sequence is called "reachable" to this shift action. The reachability of a reduce action is examined by consulting a goto graph, which is reconstructed from an LR table.

Using the procedure provided in Fig. 10, we can determine the deletable reduce actions that lead to the "deletable" shift actions.

```
for each reduce action R in each entry of LR table {
   if (every shift action which R is "reachable" to,
     has already been marked "deletable") {
        mark R "deletable";
   }
}
```

Fig. 10 Checking all reduce actions with goto graph

Fig. 11 illustrates a part of the goto graph reconstructed from the LR table in Fig. 4.

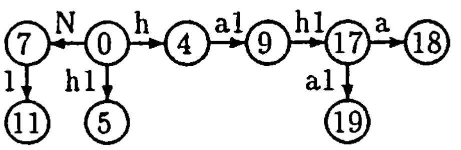

Fig. 11 A part of the goto graph from Fig. 4

Consider re7 in state 19 with a lookahead symbol "d2" in Fig. 4. According to the above goto graph, the parser will transfer to state 18 after re7, applying rule 7 (a → al) in state 19. Note: the lookahead symbol, "d2", remains the same during the reduce action. Thus, in state 18, the next reduce action becomes re2 (N → h al hl a) with the same lookahead symbol "d2".

After re2, the parser will transfer to state 7, since from state 18 we can traverse the goto graph in reverse such as "a", "hl", "al", "h" and reaches state 0 from where it transfers to state 7 by shifting N. In state 7, with the same lookahead symbol "d2", we find that sh12 has already been marked "deletable" by the substep (c), and, thus, re7 (in state 19) and re2 (in state 18) are "reachable" to sh12 (in state 7). As these reduce actions always end with sh12, we will mark both re7 and re2 "deletable".

In Fig. 4, all the deletable reduce actions found by this substep are marked (d).

(e). Deletable shift actions whose predecessors are goto actions

In this substep, we incorporate the context constraints into the shift action whose predecessors are goto actions.

If a sequence of the reduce actions reaches a specific shift action, the reduce action that finally appears in the sequence is called "immediately reachable" to the shift action.

To the shift action whose predecessors are goto actions, if all the reduce actions, which are "immediately reachable" to this shift action, are marked "deletable", this shift action should be marked "deletable". Fig. 12 shows this procedure.

```
for each shift action S in each entry of LR table {
   if (all the reduce actions which are
     "immediately reachable" to S
     have been marked "deletable") {
        mark S "deletable";
   }
}
```

Fig. 12 Checking the shift actions whose predecessors are goto actions

Unfortunately, this case does not happen in Fig. 4.

(f). The other deletable actions

After above substeps, if there is a state whose predecessors are "deletable" shift actions, all the actions in this state should be marked "deletable".

For example, in Fig. 4, re11 in state 23 with a lookahead symbol "d1" is marked "deletable" because its preceding action is sh23 in state 21 with a lookahead symbol "i1", which has already been marked "deletable" by the substep (b). In Fig. 4, all the deletable reduce actions found by this substep are marked (f).

(g). Compression of the LR table

To reduce the size of a table, we compress the LR table by:

(1) Delete all the actions that are marked "deletable",
(2) Compress all the empty states with no action,
(3) Compress all the columns (lookahead symbols) that have no action.

```
CPM()
{
   /* initialization */
      substep (a) and (b);
   /* constraint propagation */
   while(1) {
      substep (c) to (f);
      if(no new "deletable" action comes out)
        break;
   }
   /* compress LR table */
      substep (g);
}
```

Fig. 13 A top level procedure of CPM

A top level procedure of CPM is provided in Fig. 13.

The above procedure enables us to introduce the phoneme-context-dependence into the phones at word boundaries and solve the word juncture problem.

Fig. 14 is the modified allophone-based canonical LR(MCLR) table from Fig. 4 after applying CPM.

	ACTION						GOTO							
state	h1	a1	ch2	i2	d1	$	h	a	ch	i	d	N	BE	S
0	sh5		sh3				4		1			7		25
1				sh8										
3				re10										
4		sh9												
5		re5												
7					sh11						10		13	
8			sh21											
9	sh17													
10		sh15						14						
11		re13												
13						re1								
14						re4								
15						re7								
17		sh19						18						
18					re2									
19					re7									
21				sh24						22				
22					re3									
23					re12									
25						acc								

Fig. 14 The modified canonical LR(MCLR) table after CPM

4 The Effects of CPM

Compared with the table of Fig. 4, the lookahead symbols in Fig. 14 include only "h1", "a1", "ch2", "i2", and "d1". The allophones at word boundaries have been uniquely determined, since the allophone connection matrix given in Fig. 1 has very strong constraints. Generally, the allophones at a word boundary are only restricted by CPM, but not uniquely determined before seeing the next phones. This very simple example suggests how the word juncture problem can be solved by CPM.

For a task with 64 CFG rules, 120 lexical rules, the canonical LR (CLR) tables before and after applying CPM are compared by the number of states and actions. The CFG rules, dictionary, and allophone models we used are the same as in [3], which have already been used in a speech recognition system.

Table 1 shows the case of 128 allophones. After CPM, the number of states, shift actions, and reduce actions decreases to 61.4%, 38.6%, and 11% of the original canonical LR table, respectively.

Table 1 Comparison of canonical LR tables before and after CPM (128 allophones)

	state	shift	reduce	goto
CLR (before CPM)	1775	2290	10206	425
MCLR (after CPM)	1090	885	1130	425

Table 2 shows the case of 1024 allophones. After CPM, the number of states, shift actions, and reduce actions decreases to 11%, 4.9%, and 0.2% of the original canonical LR table, respectively. The more allophones, the more the table size decreases.

Table 2 Comparison of canonical LR tables before and after CPM (1024 allophones)

	state	shift	reduce	goto
CLR (before CPM)	11157	21057	701370	425
MCLR (after CPM)	1231	1026	1328	425

Reduction of the reduce and shift actions implies that the allophone predictions in speech recognition becomes more accurate.

5 Discussions and Conclusions

We have proposed a new method to generate an allophone-based LR table that solves the word juncture problem and enables to predict the allophones precisely in speech recognition. In this method, by introducing allophone rules into CFG and lexical rules, an LR table is generated, then, on the basis of an allophone connection matrix, the LR table is modified by applying CPM. Advantages of our approach can be summarized as follows:

(1) No need to modify the existing LR table generation algorithms.

(2) By introducing an allophone connection matrix and applying CPM, a large number of actions and states can be deleted, which results in enormous reduction of the table size and solves the word juncture problem. Furthermore, the reduction of actions and states provides us with accurate allophone predictions in speech recognition.

(3) The connection constraint between two adjacent allophones is incorporated in an LR table, and the

change that determines the allophones at word boundaries dynamically is not required for a GLR parsing algorithm.

One disadvantage of our method is related to (1). The number of states in a canonical LR table often explodes if the number of CFG rules increases. In order to rectify this situation, we can modify the LR table generation algorithm slightly in order to incorporate the allophone connection constraints during the generation process. In the case of 1024 allophones, the number of the states before CPM decreases to 1/5 by changing the LR table generation algorithm. Even though we can not immediately use an existing LR table generation algorithm, we should change the LR table generation algorithm if grammar and lexicon sizes become large.

The future works will include the following:

(1) Enlarging the grammar and lexical rules and allophone models and studying the efficiency of our method.

(2) Incorporating our method in an actual allophone-based continuous speech recognition system.

(3) Applying our method to stochastic context-free or context dependent grammars [5] [10] and constructing a stochastic allophone-based LR table.

Acknowledgment

The authors are grateful to Dr. S. Hayamizu and Dr. K. Itou for their valuable discussions and supports. They are also grateful to Mr. H. Nakajima for supplying us the program of generating canonical LR table.

References

[1] Aho,A.V., Sethi,R. and Ullman,J.D. *Compilers: Principles, Techniques, and Tools.* Massachusetts, Addison-Wesley, 1986.

[2] Hayamizu,S., Lee,K.F. and Hon,H.W. *Description of Acoustic Variations by Tree-Based Phone Modeling.* ICSLP90, pp.705-708, 1990

[3] Itou,K., Hayamizu,S. and Tanaka,H. *Continuous Speech Recognition by Context Dependent Phonetic HMM and an Efficient Algorithm for Finding N-best Sentence Hypotheses.* ICASSP92, pp. 21-24, 1992

[4] Kita,K., Kawabata, T. and Saitou,H. *HMM Continuous Speech Recognition Using Predictive LR parsing.* ICASSP89, 1989

[5] Lari,K. and Young,S.J. *The estimation of stochastic context-free grammars using the Inside-Outside algorithm.* Computer Speech and Language, No.4, pp. 35-56, 1990.

[6] Lee,K.F. *Automatic Speech Recognition: The Development of the SPHINX System.* Kluwer Academic Publishers, Norwell, MA, 1989

[7] Nagai,A., Sagayama,S., Kita,K. and Kikuchi,H. *Three Different LR Parsing Algorithms for Phoneme-Context-Dependent HMM Based Continuous Speech Recognition.* IEICE Trans. Inf. & Syst., vol. E76-D, No.1, pp.29-37, January, 1993

[8] Schwartz,R., Chow,Y., Kimball,O., Roucos,S. Krasner,M. and Makhoul,J. *Context Dependent Modeling for Acoustic Phonetic Recognition of Continuous Speech.* ICASSP85, 1985

[9] Tanaka,H., Tokunaka,T. and Aizawa,M. *Integration of Morphological and Syntactic Analysis Based on LR Parsing Algorithm.* International Workshop on Parsing Technologies, Tilburg, pp.101-109, 1993

[10] Wright,J.H. *LR Parsing of Probabilistic Grammars with Input Uncertainty for Speech Recognition.* Computer Speech and Language, vol.4, pp.297-323, 1990