Interactive Speech Understanding

Hiroaki Saito
Dept. of Mathematics
Keio University
Yokohama, 223, JAPAN
E-mail: hxs@nak.math.keio.ac.jp

Abstract

This paper introduces a robust interactive method for speech understanding. The generalized LR parsing is enhanced in this approach. Parsing proceeds from left to right correcting minor errors. When a very noisy portion is detected, the parser skips that portion using a *fake* nonterminal symbol. The unidentified portion is resolved by re-utterance of that portion which is parsed very efficiently by using the parse record of the first utterance. The user does not have to speak the whole sentence again. This method is also capable of handling unknown words, which is important in practical systems. Detected unknown words can be incrementally incorporated into the dictionary after the interaction with the user. A pilot system has shown great effectiveness of this approach.

1 Introduction

It has been continuously mentioned that some kind of language knowledge is essential in good-quality speech understanding. Until recently, however, most research has focused mainly on word recognition and one of the excellent recognition systems built to date is *Sphinx* developed by Lee [7]. Although Sphinx attained an excellent word accuracy of 96 % on a 997-word task, its sentence recognition accuracy drops significantly due to its use of only a statistical trigram grammar.

There have been a few attempts to integrate a speech recognition device with a natural language understanding system. Hayes *et al.* [3] adopted technique of *case frame instantiation* to parse a continuously spoken English sentence in the form of a *word lattice* (a set of word candidates hypothesized by a speech recognition module) and produce a frame representation of the utterance. The case frame parsing has been pursued by Poesio *et al.* [8] and Giachin *et al.* [2] for instance.

Meanwhile, a compiler-oriented shift-reduce LR parsing technique has been used for speech recognition recently due to its no-backtracking table-driven efficiency [12, 10, 6]. Because the parsing proceeds from left to right pruning low-probability partial-parses, the correct parse can not be obtained if the parsing fails to find the correct path in the beginning. Moreover, it is sometimes difficult to handle the very noisy input, especially the input with missing words. Thus an LR parser sometimes yields totally incorrect but syntactically-sound hypotheses or no hypotheses at all. This weakness is occasionally cited to demonstrate superiority of the parsing method using much simpler bigram or trigram grammars in which the recovery in the middle of the input can be done at ease. In this paper, we describe a method of enhancing the generalized LR (GLR) parsing towards interactive speech understanding.

Section 2 describes the enhanced GLR parsing. Section 3 describes the robustness of the parser and presents an interactive method to resolve the unclear portion of the input and unknown words. Section 4 experiments the effectiveness of the technique in parsing spoken sentences. Finally the concluding remarks are given in Section 5.

2 Enhanced GLR Parsing for Speech Understanding

In this section, the GLR parsing method is described first. Then some techniques which enhance the robustness are described.

2.1 Background: GLR Parsing

The LR parsing technique was originally developed for the compilers of programming languages [1] and has been extended for natural language processing [11]. The GLR parsing analyzes the input sequence from left to right with no backtracking by looking at the parsing table constructed from the context-free grammar rules in advance. An example grammar and its parsing table are shown in Figure 1 and Figure 2 respectively.

Entries "s n" in the action table (the left part of the table) indicate the action "shift one word from the 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". "$" in the action table is the end-of-input symbol. The goto table (the right part of the table) decides to which state the parser should go after a reduce action. The LR parsing table in Figure 2 is different from regular LR tables utilized by the compilers in that there are multiple entries, called *conflicts*, on the rows of state 11 and 12. While the encountered entry has only one action, parsing proceeds exactly the same way as the regular LR parsing. In case there are multiple actions in an entry, all the actions are executed with the *graph-structured stack* [11].

```
-------------------------
(1)  S   -->  NP  VP
(2)  S   -->  S  PP
(3)  NP  -->  n
(4)  NP  -->  det  n
(5)  NP  -->  NP  PP
(6)  PP  -->  prep  NP
(7)  VP  -->  v  NP
-------------------------
```

Figure 1: Example CFG Rules

2.2 GLR Parsing for Erroneous Sentences

The original GLR parsing method was not designed to handle ungrammatical sentences. This feature is acceptable if the domain is strictly defined and input sentences are correct at all times. Unfortunately, accuracy of speech recognition is not 100%. Common errors in speech recognition are insertions, deletions (missing words), and substitutions. Some techniques have been developed to handle erroneous sentences for the GLR parsing [12, 10].

	<Action Table>					<Goto Table>			
	det	n	v	prep	$	NP	PP	VP	S
0	s3	s4				2			1
1				s6	acc		5		
2			s7	s6			9	8	
3		s10							
4			r3	r3	r3				
5				r2	r2				
6	s3	s4				11			
7	s3	s4				12			
8				r1	r1				
9			r5	r5	r5				
10			r4	r4	r4				
11			r6	r6,s6	r6		9		
12				r7,s6	r7		9		

Figure 2: GLR Parsing Table

- The action table can be looked up in a predictive way to handle a missing word. Namely, a set of possible terminal symbols $\{T_i\}$ at State i can be missing word candidates.
- This way of using the action table is also useful to handle substitution and insertion errors. I.e., the table can tell which part of the input should be replaced by a specific symbol or ignored.

The parser explores every possibility in parallel[1].

2.3 Gap-filling Technique

The techniques described in the previous section can not handle such a big noise as two consecutive missing words. To cope with this, the gap-filling technique [9] is presented here.

In the gap-filling GLR parsing, the goto table is consulted just the same way as the action table, in addition to its regular usage. Namely, at state s_i which is expecting shift action(s), the parser also consults the goto table. If an entry m exists along the row of state s_i under the column

[1] In practice, pruning is incorporated to reduce search by using the likelihood attached to each word in the speech hypotheses.

labeled with nonterminal D, the parser shifts D onto the stack and goes to state m. Note that no input is scanned when this action is performed. When the input is incomplete, the parser produces hypotheses with a *fake* nonterminal at the noisy position.

We show an example of parsing an incorrectly recognized sentence "we cut sad with a knife" using the grammar in Figure 1[2] and the LR table in Figure 2.[3] At the initial state 0, the goto table tells that the nonterminals NP and S can be shifted using the gap-filling technique. Although the first word "we" (noun) is expected at state 0, these fake nonterminals are created (Figure 3) in case "we" is an incorrectly recognized word. The new states for the fake nonterminals NP and S are 2 and 1, respectively. The goto table tells that fake nonterminals PP and VP can be placed at state 2. In this case, however, we do not create these nonterminals, because two fake nonterminals rarely need to be placed adjacently in practice. No further fake nonterminal is attached to the fake nonterminal S for the same reason.

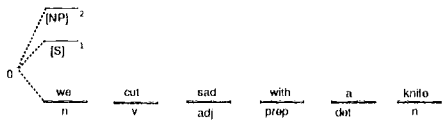

Figure 3: Parse Trace

In parsing the third word "sad", a fake nonterminal [NP] to word "cut" keeps the correct path (Figure 4).

Parsing continues in this way and the final situation is shown in Figure 5. As a result, the parser finds two successful parses:

```
(n (v ([NP] (prep (det n)))))

((n (v [NP])) (prep (det n)))
```

Namely, the parser finds out that the third word is incorrect and must be the word(s) in NP category.

[2]The terminal symbols of this grammar are grammatical category names called *preterminals*. A lexicon should be prepared to map an actual word to its preterminal.

[3]The techniques in the previous section are enough for parsing this erroneous sentence. We use this example only for describing the gap-filling technique.

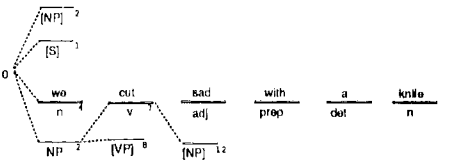

Figure 4: Parse Trace (cont'd)

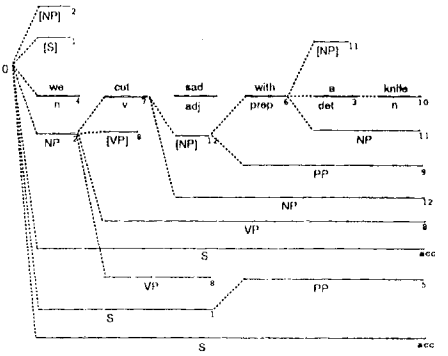

Figure 5: Parse Trace (complete)

3 Interactive Speech Understanding

In this section, the robustness of the GLR parser with various error-recovery techniques (especially the gap-filling technique) against a noisy input is described. Then an interactive way to resolve the unidentified portion is presented.

3.1 Resolving Unidentified Portion

The gap-filling technique enhances the robustness of the GLR parsing in handling a noisy input as follows:

- A fake nonterminals fills big missing constituents of the input which would yield no hypotheses without the gap-filling function.
- The gap-filling function enables an LR parser to perform reduce actions only when the action creates a definite high-score nonterminal. The fake nonterminal is likely to be either an insertion or an unknown word.

A gap filled with a fake nonterminal can be resolved by reanalysis of the input under the constraint that that portion of the input should yield the specific nonterminal. This top-down reanalysis would be effective against the genuinely bottom-up GLR parsing. In practice, however, a more reliable way is to ask the user to speak only the missed portion. In the previous example, only the portion of [NP] should be spoken again.

The parser can analyze the re-utterance efficiently as follows:

1. The parser keeps the parse record of the first input.
2. The parser starts parsing the new input just where the fake nonterminal was created.
3. The parsing ends when the same-name real nonterminal symbol is created out of the reutterance.

3.2 Handling Unknown Words

If the reutterance can not be parsed correctly even by the reutterance, the unidentified portion is likely to contain an unknown word. Finding an unknown word by a specific nonterminal symbol enables the interactive grammar augmentation as the following, for instance.

```
The parser can not identify the following
portion of your input.

We cut [NP] with a knife

If this is a new word in the category of [NP],
a rule
NP --> (recog. result of the 2nd utterance)
will be added to the grammar.    Is this ok?
```

Handling unknown words is important in natural language processing. For example, Kamioka *et al.* [5] proposed a mechanism which parses a sentence with unknown words using Definite Clause Grammars. The efficient gap-filling technique of handling unknown words is quite useful in practical systems and enhances the robustness of the GLR parsing greatly.

When an unknown word W_{new} is detected, the word should be incorporated into the system. If the grammar is separated from the lexicon, the word can be easily added to the dictionary. If the grammar contains the lexicon, the LR table should be augmented incrementally in the following way.

1. For each state s_i which has an entry under the column of the nonterminal $D_{(fake)}$ in the goto table, add shift action "s m" (m is the new state number) for W_{new}. (If W_{new} consists of such multiple words as "*get rid of*", a new state should be created for each element of the words.)
2. Add reduce action "r p" (p is the new rule number) for all the terminals on the row of state m.

Before we close this section, we should consider side effects of the gap-filling technique. It is true that putting fake nonterminals expands search. Thus, some side effect might appear if the accuracy of input is not good. Namely, input should be good enough to produce distinct fake nonterminals and real nonterminals. Although it is difficult to analyze this phenomenon theoretically, the following natural heuristics can minimize the search growth.

- Two consecutive fake nonterminals are not allowed as shown in the previous section.
- When a word (W_i) can be shifted to both a fake nonterminal D_{fake} and a same-name real nonterminal D_{real}, only D_{real} should be valid.
- When D_{fake} and D_{real} can be bundled using the *local ambiguity packing* [11] technique, discard D_{fake}.

4 Experiments: Parsing Spoken Sentences

We evaluated effectiveness of the enhanced GLR parsing by spoken input. We used a device which recognizes a Japanese utterance and produces its phoneme sequence [4]. The parser we used is based on the GLR parser exploring the possibilities of substituted/inserted/deleted phonemes [10] by looking up the confusion matrix, which was constructed from the large vocabulary data. The confusion matrix is also used to assign the score to each explored phoneme, because the recognition device gives neither the alternative phoneme candidates nor the likelihood of hypothesized phonemes. The gap-filling function is incorporated into the parser in the following experiments. Parsing a phoneme sequence might sound less popular than parsing a word lattice in speech

recognition. Because the parser builds a lattice dynamically in parsing the sequence from left to right using a CFG which contains the dictionary, no *static* lattice is necessary.

125 sentences (five speakers pronounced 25 sentences) were tested in the domain called "conversation between doctors and patients." 111 sentences were parsed correctly [88.8 %] (the correct sentence was obtained as the top-scored hypothesis). 14 failed sentences can be classified into three groups:

(i) 4 sentences were parsed as the top-scored hypotheses with fake nonterminals. Thus the parser asked the user to speak the unidentified portion again.

(ii) 6 sentences were parsed incorrectly in that the correct sentence did not get the highest score mainly because the incorrect nonterminal had a slightly higher score than the correct one. In this case, both the closely-scored correct and incorrect nonterminals are packed into one nonterminal using the *local ambiguity packing* technique in an efficient implementation. In this situation the parser should ask the user to speak only that unclear portion in the same way as in (i) instead of producing a barely top-scored hypothesis. In the current implementation the parser asks the user which word is the correct one.

(iii) 4 sentences were pronounced very badly. The user has to speak the whole sentence again.

5 sentences with unknown words were also tested. In all cases, the unknown word was detected.

This result shows that interactive partial reutterance is very effective both for error-recovery and for detection of unknown words.

5 Concluding Remarks

We presented a robust interactive approach for speech understanding. The GLR parsing method was enhanced to recover errors and to skip a very noisy portion. These techniques remedy all-or-nothing-ness of the CFG-based LR parsing. The skipped portion is represented by a fake nonterminal which is resolved by re-utterance. An unknown word is also detected by a fake nonterminal and is incorporated into the dictionary incrementally through interaction with the user. Experiments in parsing a Japanese phoneme sequence have shown a great effectiveness of this interactive approach.

References

[1] Sethi R. Aho, A. V. and J. D. Ullman. *Compilers.* Addison Wesley, 1986.

[2] E. Giachin and C. Rullent. Robust Parsing of Severely Corrupted Spoken Utterances. In *Proceedings, 12th International Conference on Computational Linguistics (COLING)*, Budapest, Hungary, August 1988.

[3] Hauptmann A. G. Carbonell J. G. Hayes, P. J. and M. Tomita. Parsing spoken language: A semantic caseframe approach. In *Proceedings, 11th International Conference on Computational Linguistics (COLING)*, West Germany, August 1986. Bonn.

[4] Morii S. Hoshimi M. Hiraoka, S. and K. Niyada. Compact isolated word recognition system for large vocabulary. In *Proceedings, IEEE-IECEJ-ASJ International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Tokyo, April 1986.

[5] T. Kamioka and Y. Anzai. Analysis of sentences including unknown words by hypothesis generation mechanism. *Journal of Japanese Society for Artificial Intelligence, Vol.3 No.5*, pages 627–638, September 1988. [In Japanese].

[6] Kawabata T. Kita, K. and H. Saito. HMM Continuous Speech Recognition Using Predictive LR Parsing. In *ICASSP*, May 1989.

[7] K.F. Lee. *Large-Vocabulary Speaker-Independent Continuous Speech Recognition: The SPHINX System.* PhD thesis, Computer Science Department, Carnegie Mellon University, April 1988.

[8] M. Poesio and C. Rullent. Modified Caseframe Parsing for Speech Understanding Systems. In *Proceedings, 10th International Joint Conference on Artificial Intelligence (IJCAI)*, Milan, August 1987.

[9] H. Saito. Gap-filling LR Parsing for Noisy Spoken Input: Towards Interactive Speech Recognition. In *Proceedings, International Conference on Spoken Language Processing (ICSLP)*, Kobe, Japan, November 1990.

[10] H. Saito and M. Tomita. Parsing Noisy Sentences. In *Proceedings, 12th International Conference on Computational Linguistics (COLING)*, Budapest, Hungary, August 1988.

[11] M. Tomita. *Efficient Parsing for Natural Language.* Kluwer Academic Publishers, Boston, MA, 1985.

[12] M. Tomita. An efficient word lattice parsing algorithm for continuous speech recognition. In *Proceedings, IEEE-IECEJ-ASJ International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Tokyo, April 1986.