STUDIA PHONOLOGICA XXII (1988)

Consideration on Syntactic Analyses for a Speech Understanding System

Seiichi NAKAGAWA and Yoshihisa OHGURO

SUMMARY

In this paper, we compared the left-to-right & top-down parsing strategy with the island-driven & botom-up strategy by using a simulated phoneme recognizer. The both strategies adopted the beam search. The syntactic constraints were represented by a context-free grammar. The word lattice for an utterrance was generated by a word spotting algorithm from an ambiguous phoneme sequence. These two parsing strategies were used for finding the best word sequence from the word lattice. We evaluated them through three different tasks (small, combined and large).

From the parsing results by the two strategies, we can conclude in the following: (1) the recognition accuracy was almost the same for both strategies, (2) the left-to-right & top-down parsing strategy was superior to the island-driven & bottom-up strategy in terms of the processing time.

And also we found that the sentence recognition rate was not so sensitive on the vocabulary size, or on the combined task in comparison with a single task.

However, when the initial part of an utterance was noisy, the island-driven strategy became superior to the left-to-right strategy. Finally, we found that our proposed parsing machanism for missing function words was effective.

1. INTRODUCTION

In speech understanding systems, there are two basic control strategies for the syntactic analyses. One is a left-to-right parsing control strategy, and this strategy has been used for a syntactic analysis of text inputs or speech inputs. The standard parsing algorithms are based on Earley's algorithm (top-down), 'CYK' algorithm (bottom-up), and Augmented Transition Network grammar.

The other control strategy is an island-driven strategy. This strategy was attractive for speech understanding systems, because candidate words obtained

Seiichi NAKAGAWA (中川聖一): Associate Professor, Department of Information and Computer Sciences, Toyohashi University of Technology, Toyohashi, 440 Japan.
Yoshihisa OHGURO (大黒慶久): Graduate Student, Department of Information and Computer Sciences, Toyohashi University of Technology, Toyohashi, 440 Japan.
The authors have done the research under the direction of Dr. Shuji Dohshita, Professor of Information Science, Kyoto University.

from speech input were not always correct. Indeed, the HWIM speech understanding system developed in BBN adopted this strategy by using the Augmented Transition Network grammar[1]. This system begins the parsing from most reliable seeds obtained by the initial scan of utterance.

There are also the other basic choices for the parsing strategy. They are a backtracking search versus a parallel search. In speech understanding systems, the most optimal word sequence should be found in a word lattice because the detected words were not perfect and had scores of reliability. In such a case, the paralle search is suitable and is usually implemented in a beam search[2][3]. The search is more efficient than a best-first (A*) search[4].

Levinson studied on the effects of syntactic analysis on speech recognition by means of a computer simulation[5]. He found that an acoustic word error rate of 10 % was reduced to 0.2 % after syntactic analysis, resulting in a sentence error rate 1 % for a 127-word vocabulary task. Recently, Miller and Levinson tested the effectiveness in a similar simulation technique using a 1040 vocabulary and a context-free covering grammar[6].

In this study, we compared the left-to-right & top-down parsing strategy with the island-driven & bottom-up strategy by using a simulated phoneme recognizer. The both strategies adopted the beam search. The syntactic constraints were represented by a context-free grammar. The word lattice for an utterrance was generated by a word spotting algorithm from an ambiguous phoneme sequence. The input of parser consists of a word lattice of candidate or spotted words, which are identified by their begin and end times, and the score of the acoustic-phonetic match. Recently, Ward et al. have also studied on a similar comparison[7]. They found that the island-driven parser produced parses with a higher percentage of correct words than the left-to-right parser in all cases considered. However they did not use the grammatical constraints expressed in a context-free grammar, but trigram models of sentences of lexical and semantic labels. Their evaluation criterion was the rate of correct word. Our criterion was the rate of correct sentence. Therefore, our conclusion was not comparable with their results. First, we describe the simulator of a phoneme recognizer and two parsing strategies which were used for finding the best word sequence from the word lattice.

2. Simulator of a Phoneme Recognizer and Word Spotting

2. 1 Simulator of a Phoneme Recognizer

Since we would like to know the effectiveness of syntactic analysis for various qualties of phoneme recognizers, we simulated phoneme recognizers. So we generate ambiguous phoneme sequences including substitution errors, insertion errors and omission errors for a correct phoneme sequence corresponding to an input sentence on the basis of assumed confusion matrix of phoneme recognizers.

Fig. 1 shows the confusion matrix, where the parameter P denotes the phoneme classification correct rate. We also assumed that both the insertion and omission errors are 5 % for all phonemes. In our simulation, we set the parameter P to 60 % or 80 %.

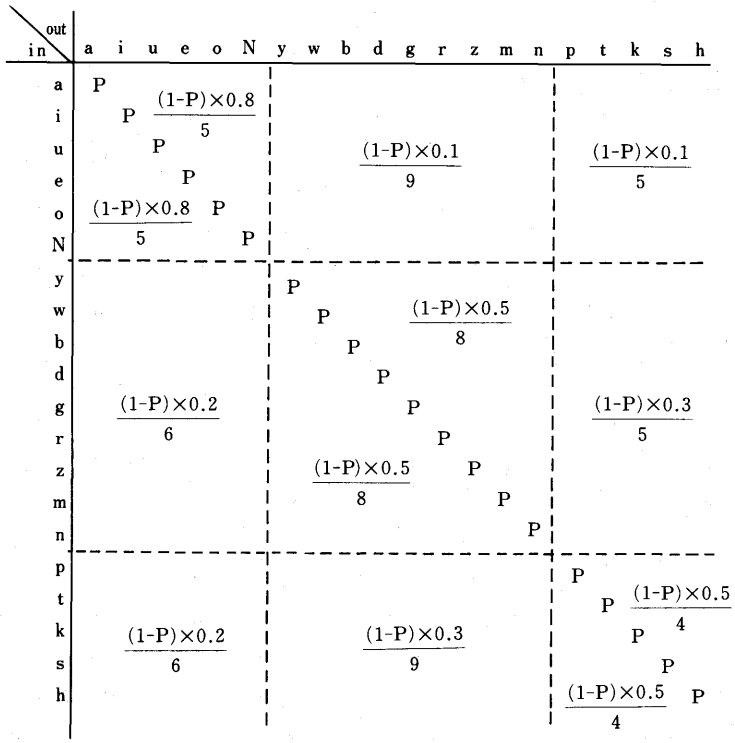

Fig. 1. Confusion matrix between phonemes (P denotes the phoneme recognition rate)

2. 2 Word Spotting Algorithm

For speech understanging systems, the problem of detecting and locating a specified word in continuous speech has been considered. In 1975, Sakai and Nakagawa proposed a word spotting algorithm by using a dynamic time warping algorithm[2]. It was applied to the phoneme sequences and detected the key words in a given task. It was very difficult to spot functional words from a classified phoneme sequence, because the phoneme sequence contains insertion, omission or substitution errors. Christiancen and Rushforth applied a similar algorithm to spectral sequences[8], but their method was suboptimal in the application of dynamic programming. In this paper, we modify our previous method for working on a phoneme sequence generated by a confusion matrix.

〈Notation〉

$A=a_1, a_2, \cdots\cdots, a_i, \cdots\cdots, a_I$:

denotes the ambiguous phoneme sequence as the output of a phoneme recognizer.

$B=b_1, b_2, \cdots\cdots, b_j, \cdots\cdots, b_J$:

denotes the correct phoneme sequence for a lexical word.

$P(i, j)$:

denotes the logarithm of confusion probability between a_i and b_j.

$Q(i, j)$:

denotes the maximum likelihood in terms of m for (maximum cumulative likelihood between $a_m, a_{m+1}, \cdots, a_i$ and $b_1, b_2, \cdots, b_j$).

$B(i, j)$:

denotes the argument of maximum in terms of m for (maximum cumulative likelihood between $a_m, a_{m+1}, \cdots, a_i$, and $b_1, b_2, \cdots, b_j$).

P_I: the insertion probability, 5 %, for all phonemes.

P_0: the omission probability, 5 %, for all phonemes.

P_S: the substitution probability, that is, $1.0-P_I-P_0$.

If we could calculate $Q(i, j)$ for the i-th phoneme in the input sequence and the lexical phoneme sequence, the word spotting problem would be solved. That is, if $Q(i, J)$ satisfies the threshold for word detection, the location can be regarded as $B(i, J)\sim i$ in the input phoneme sequence.

$Q(i, J)$ and $B(i, J)$ are calculated by the following DP equation:

1. Initialize
   $Q(-1, j)=Q(0, j)=-1000000$ for $j=1, 2, \cdots, J$
2. Execute steps 3, 4 for $i=1, 2, \cdots, I$
3. Word boundary process (set of beginning point)
   $B(i, j)=i$
   $q1=-1000000$
   if $i>1$, $q1=\log P(i, 1)+\log P(i-1, 1)+\log P_I$
   $q2=\log P(i, 1)+\log P_S$
   $q3=\log P_0$
   $Q(i, 1)=\max\{q1, q2, q3\}$
   if $Q(i, 1)=q1$, $B(i, 1)=i-1$
4. For $j=2, 3, \cdots, J$
   $q1=q2=-1000000$
   if $i\geqq 3$, $q1=Q(i-2, j-1)+\log P(i-1, j)+\log P(i, j)+\log P_I$
   if $i\geqq 2$, $q2=Q(i-1, j-2)+\log P(i, j)+\log P_S$
   $q3=Q(i, j-1)+\log P_0$
   $Q(i, j)=\max(q1, q2, q3)$
   if $Q(i, j)=q1$, $B(i, j)=B(i-2, j-1)$
   if $Q(i, j)=q2$, $B(i, j)=B(i-1, j-1)$
   if $Q(i, j)=q3$, $B(i, j)=B(i, j-1)$
5. The likelihood of word spotting (score)$=1000+100\times Q(i, J)/J$

2.3 Word Detection Procedure

As described above, if we set the threshold for word spotting, we can detect

candidate location and likelihood for a given word[10][11]. However, in general, many similar locations are detected, e. g., the same locations except for ending position. Therefore, if the candidate locations are detected for the successive ending positions, we select only the most probable location on the basis of cumulative likelihood or the normalized average likelihood by the length of the lexical word. If this procedure is performed on all the lexical entries, we can generate a word lattice for the input utterance (ambiguous phoneme sequence).

The word spotting result is represented by the set of four items; (beginning point, end point, word name, score). Fig. 2 illustrates an example of the word spotting result.

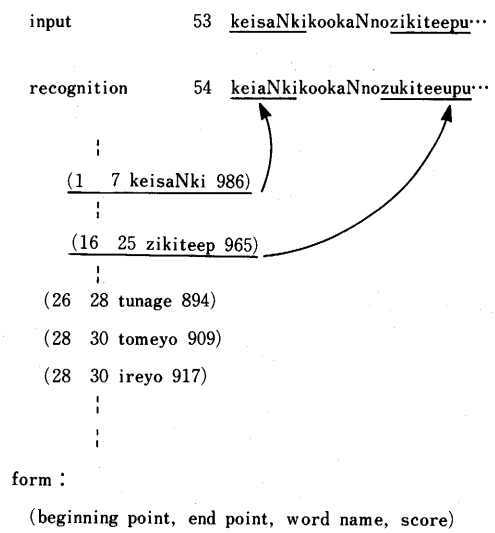

Fig. 2. An example of a word lattice.

3. REPRESENTATION OF GRAMMAR AND WORD PREDICTOR

The syntactic knowledge in terms of the "task" is given by the grammar. This grammar is represented by the context-free grammar as shown in Figure 3. The variable with the affix "@" shows that it is a nonterminal symbol, and the variable with "$\ast$" a word class (a kind of nonterminal symbols). The word class contains the set of words with the same syntactical category. The numbers of the row and column define the position of a production rule. They will be used for the parsing algorithm described in the next section.

In the LITHAN speech understanding system[2], we proposed an efficient context-free parsing algorithm which is similar to Earley algorithm[9]. Fig. 4 shows an example of a part of a parsing table by Earley algorithm, where the input is "MARY WILL PLAY..." and the grammar is given in Fig. 3. This algorithm makes a parse tree table (set of states) in the fashion of left-to-right. The

	0		1	2	3	4	5
8	@S	→	@NP	@VP			
16	@S	→	@NP	⋇AUX	@VP		
24	@S	→	⋇AUX	@VP			
32	@NP	→	⋇DET	@NP2			
40	@NP	→	@NP2				
48	@NP2	→	⋇ADJ	@NP2			
56	@NP2	→	@NP3				
64	@VP	→	⋇VERB				
72	@VP	→	⋇VERB	@PP			
80	@VP	→	⋇VERB	@NP			
88	@PP	→	⋇PREP	@NP			
96	@NP3	→	@NP3	@PP			
104	@NP3	→	⋇NOUN				

word class

⋇NOUN	→	JOHN
⋇NOUN	→	MARY
⋇NOUN	→	MAN
⋇NOUN	→	I
⋇NOUN	→	TENNIS
⋇NOUN	→	GAME
⋇AUX	→	WILL
⋇AUX	→	CAN
⋇VERB	→	KNOW
⋇VERB	→	PLAY
⋇VERB	→	PLAYED
⋇DET	→	THE
⋇DET	→	A
⋇ADJ	→	BIG
⋇ADJ	→	YOUNG
⋇PREP	→	BY
⋇PREP	→	OF
⋇PREP	→	WITH

Fig. 3 An Example of Context-Free Grammar.

predictor and completer from states are not efficient, because it needs many times of searches in the table. Therefore, we modify this algorithm in the following.

Let the position of a production rule be represented by the number[11]. For example, the number "8" denotes @S, "9" @NP, and "19" @VP in the above grammar. The basic problem is formalized as follows: Which words are predicted as the succeeding words, when a partial sentence is given? For example, when the partial sentence "MARY WILL PLAY" is given, which words could be appear in the right hand side? In this cace, the partial sentence is derived by the following production rules: @S → @NP ⋇AUX @VP → @NP2 ⋇AUX @VP → @NP3 ⋇AUX @VP → ⋇NOUN ⋇AUX @VP → MARY ⋇AUX @VP → MARY WILL @VP → MARY WILL ⋇VERB, MARY WILL ⋇VERB @PP or MARY WILL ⋇VERB @NP → MARY WILL PLAY, MARY WILL PLAY @PP or MARY WILL PLAY @NP. Therefore the succeeding words could be

S_0 (X_1=MARY)	@S	→ · @NP	⋇AUX		@VP	.0
	@NP	→ · @NP2				.0
	@NP2	→ · @NP3				.0
	@NP3	→ · ⋇NOUN				.0
	⋇NOUN	→ · MARY				.0
		⋮				
S_1 (X_2=WILL)	⋇NOUN	→ MARY ·				.0
	@NP3	→ ⋇NOUN ·				.0
	@NP2	→ @NP3 ·				.0
	@NP	→ @NP2 ·				.0
	@S	→ @NP ·	⋇AUX		@VP	.0
	⋇AUX	→ · WILL				.1
		⋮				
S_2 (X_3=PLAY)	⋇AUX	→ WILL ·				.1
	@S	→ @NP	⋇AUX ·		@VP	.0
	@VP	→ · ⋇VERB				.2
	@VP	→ · ⋇VERB		@PP		.2
	@VP	→ · ⋇VERB		@NP		.2
	⋇VERB	→ · PLAY				.2
		⋮				
S_3 (X_4= ?)	⋇VERB	→ PLAY ·				.2
	@VP	→ ⋇VERB ·				.2
	@VP	→ ⋇VERB ·		@PP		.2
	@VP	→ ⋇VERB ·		@NP		.2
	@S	→ @NP	⋇AUX	@VP ·		.0
	@PP	→ · ⋇PREP	@NP			.3
	@NP	→ · @NP2				.3
		⋮				

Fig. 4 An example of parsing table by Earley algorithm.

predicted from @PP and @NP. "BY", "OF", "WITH", "THE", "A", "BIG", "YOUNG", "JOHN", "MARY", "MAN", "I", "TENNIS" and "GAME" are predicted.

We can memorize the application order of the procuction rules by the sequence of positions in the grammar. For the above example, "MARY WILL PLAY" is derived by the sequences "16" → "17" → "17" 40" → "17 41" → "17 41 56" → "17 41 57" → "17 41 57 104" → "17 41 57 105" → prediction of ⋇NOUN → "18" → prediction of ⋇AUX → "19" → "19 64", "19 72" or "19 80" → "19 65", "19 73" or "19 81" → prediction of ⋇VERB. For convenience sake, we call this sequence a "grammar path". The recursive algorithm for parsing or prediction is given below:

1. Enter the given grammar path into "path list".
2. If the path list is empty, then stop. Otherwise, select a grammar path from

the path list. Add 1 to the number of the most right hand side in the grammar path. This number indicates the next position to be processed in the grammar.

- If the variable of the position is a terminal symbol, predict the word (terminal symbol) and return the grammar path. Then go to step 2.
- If the variable of the position is a word class with the affix "*", predict the set of words for the word class and retern the grammar path. Then go to step 2.
- If the variable of the position is a nonterminal with the affix "@", the production rules with the same nonterminal at the left hand side are precicted, that is, the head positions of these rules are concatenated at the most right hand side of the grammar path. Enter these paths into the path list. Then go to step 2.
- If the variable of the position is empty, eliminate the number of the most right hand side in the grammar path and enter this path into the path list. then go to step 2.

In this procedure, we should pay attention to the representation of left recursion in a production rule, e. g., @NP3 → @NP3 @PP → @NP3 @NP3 @PP → ···. Therefore we must restrict the application of such a rule. Although the times of the application are generally restricted, we restrict the length of grammar path.

In the above example, "MARY WILL PLAY" was derived by the three different derivations, that is, three different grammar paths. (It is called "an ambiguous grammar".) This is not efficient. Therefore, we should rewrite the grammar as shown in Figure 5. The representation of Figure 5 is efficient because a partial sentence has only one grammar path. (Such a grammar is called "an

	0		1	2	3	4	5
8	@S	→	@NP	@AUX	@VP		
16	@S	→	*AUX	@VP			
24	@AUN	→					
32	@AUN	→	*AUX				
40	@NP	→	@DEN	@NP2			
48	@DEN	→					
48	@DEN	→	*DET				
56	@NP2	→	*ADJ	@NP2			
64	@NP2	→	@NP3				
72	@VP	→	*VERB	@PN			
80	@PN	→					
88	@PN	→	@PP				
96	@PN	→	@NP				
104	@PP	→	*PREP	@NP			
112	@NP3	→	@NP3	@PP			
120	@NP3	→	*NOUN				

Fig. 5. Equivalent Representation of the Example Grammar.

unambiguous grammar".) This representation is also effective for the beam search described below, which has the limitation for the number of grammar paths memorized.

4. TIME-SYNCHRONOUS LEFT-TO-RIGHT PARSING ALGORITHM[10]

4.1 Backward Algorithm W*[10]

Next, we extend this parsing/prediction algorithm to the time-synchronous one. We already proposed the basic idea which combined the word spotting algorithm with a syntactical constrained connected word recognition algorithm. We call it "Augmented Continuous DP Matching Algorithm[11]". In the literature, the syntactic knowledge was represented by a finite state automaton or regular grammar. In this section, we extend it to a context-free grammar. Figure 6 illustrates the flow of the backward parsing algorithm W*.

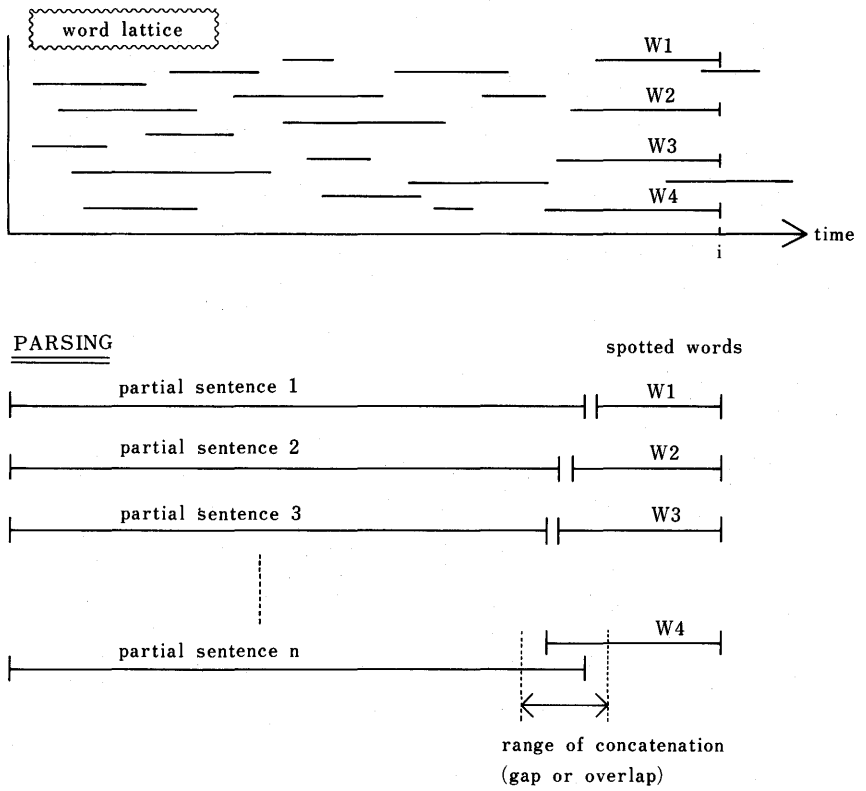

Fig. 6. A time-synchronous left-to-right parsing method (Backward Parsing Algorithm W*)

If all possible partial sentences are taken into consideration, the computation time or memory spaces will become large. Therefore we select a few best partial sentences and abandon the others. We proposed this pruning technique in the LITHAN speech understanding system[2]. In general, this search technique is well

known as "beam search". First we define the following notations:

I: length (in frames) of an input sentence.

$a_1, a_2, \cdots, a_i, \cdots, a_I$: the feature vector sequence of the input pattern.

CAND(m): the m-th word candidate detected by the word spotting algorithm.

BEG(m): the beginning frame of CAND(m).

END(m): the ending frame of CAND(m).

SCORE(m): the matching score for CAND(m), normalized by the length of the reference pattern. (The larger is the score, the better is the matching.)

NOCAND: the number of candidates detected by the word spotting algorithm.

WSEQ(i, k): the k-th best word sequence which matches with $a_1, a_2, \cdots, a_i$ of the input pattern, which satisfies the given grammatical constraints.

CSCORE(i, k): the cumulative score for matching WSEQ(i, k) with $a_1, a_2, \cdots, a_i$.

GPATH(i, k): the grammar path corresponding to the partial sentence WSEQ(i, k).

PREDICT(i, k): the set of words which are enable to appear in the right hand side of WSEQ(i, k).

BEAM: the upper limit of number of partial sentences memorized for every input frame, that is, width of beam search.

PARSER(arg): function of parsing, which has a grammar path as an argument, and returns with predicted words and the grammar paths. The parsing algorithm has already been described above.

The algorithm is the following:

〈Backward Algorithm W*〉

1. Initialization
   GPATH(0, 1)="8"
   PREDICT(0, 1) ← {predicted words} by PARSER(GPATH(0, 1))
2. Execute steps 3, 4, 5, 6 for $i=1, 2, \cdots, I$
3. $k=0$.
   Execute steps 4, 5 for all m such that END(m)=i.
4. Execute step 5 for j=BEG(m)−gap1, BEG(m)−gap1+1, ···, BEG(m)+gap2.
5. for $r=1, 2, \cdots,$ BEAM
   if CAND(m)∈PREDICT(j, r), $k=k+1$.
   GPAGH(i, k) ← {grammar path} by PARSER(GPATH(j, r)) which predicted CAND(m).
   WSEQ(i, k)=WSEQ(j, r)⋊CAND(m).
   CSCORE(i, k)=CSCORE(j, r)+SCORE(m)×($i-j$)
   If PARSER(GPATH(j, r)) generates plural GPATH(i, k) which memorize all the plural GPATH(i, k), WSEQ(i, k) and CSCORE(i, k).
   Note that the contents of WSEQ(i, k) and CSCORE(i, k) for all the plural memories are the same, respectively. The operator "⋊" denotes the concatenation of a word sequence at the first operand and a word at the second

operand.

6. Sort CSCORE(i, k) and select only the top BEAM partial sentences, cumulative scores, grammar paths, and predictive words on the basis of sorted cumulative scores, that is, WSEQ(i, r), CSCORE(i, r), GPATH(i, r), PREDICT(i, r) and PARSER(GPATH(i, r)), for $r=1, 2, \cdots$, BEAM.

If different word sequences have the same grammar path at the same frame in the input pattern, these partial sentences behave the same action hereafter. Therefore, we keep only the better sequences, and abandon the others. Inversely, if different grammar paths have the same word sequence at the same frame, these paths and sequences are kept, respectively. The gap1 and gap2 denote the searching region of beginning frame of a spotted word. This is to cope with the mis-detection of spotting. The gap2 may corespond to the overlapping region between words.

There are three kinds of word sequences. The first is a partial sentence and an incomplete sentence, where a partial sentence means that it has a succeeding word at the most right hand side. The second is a partial sentence and a complete sentence. The third is a complete sentence, but not a partial sentence. If at least one of contents of grammar paths obtained by PARSER(GPATH(i, k)) becomes empty on the way of prediction, WSEQ(i, k) is a partial sentence and a complete sentence. If all the contents of grammar paths become empty, WSEQ(i, k) is a complete sentence.

4.2 Forward Algorithm S*[10]

Strictly speaking, the required space for PREDICT(i, k) or temporary results of the PARSER becomes large. Therefore, in practice, we use commonly PREDICT(i, k) for all k in our experiments. However, this is evidently not efficient, because we must execute the PARSER more than two times for the same argument if there exists a legal word.

The algorithm W* is time-synchronous in terms of the ending frame of spotted words. We propose another efficient time-synchronous parsing algorithm S*, which is time-synchronous in terms of the ending frame of generated partial sentences. This algorithm executes the prediction of words at the right hand side of a partial sentence and the concatenation of a spotted word (candidate word) at the same time. Figure 7 illustrates the flow of the forward parsing algorithm S*. The algorithm is the following, where BREADTH(i) denotes the number of partial sentences which end at the frame i:

〈Forward Algorithm S*〉

1. Initialization.
   GPATH(0, 1)="8".
   BREADTH(0)=1, BREADTH(i)=0 for $i>0$.
2. Execute steps 3, 4, 5 for $i=0, 1, 2, \cdots, I$
3. Execute steps 4, 5 for $r=1, 2, \cdots$, BREADTH(i)

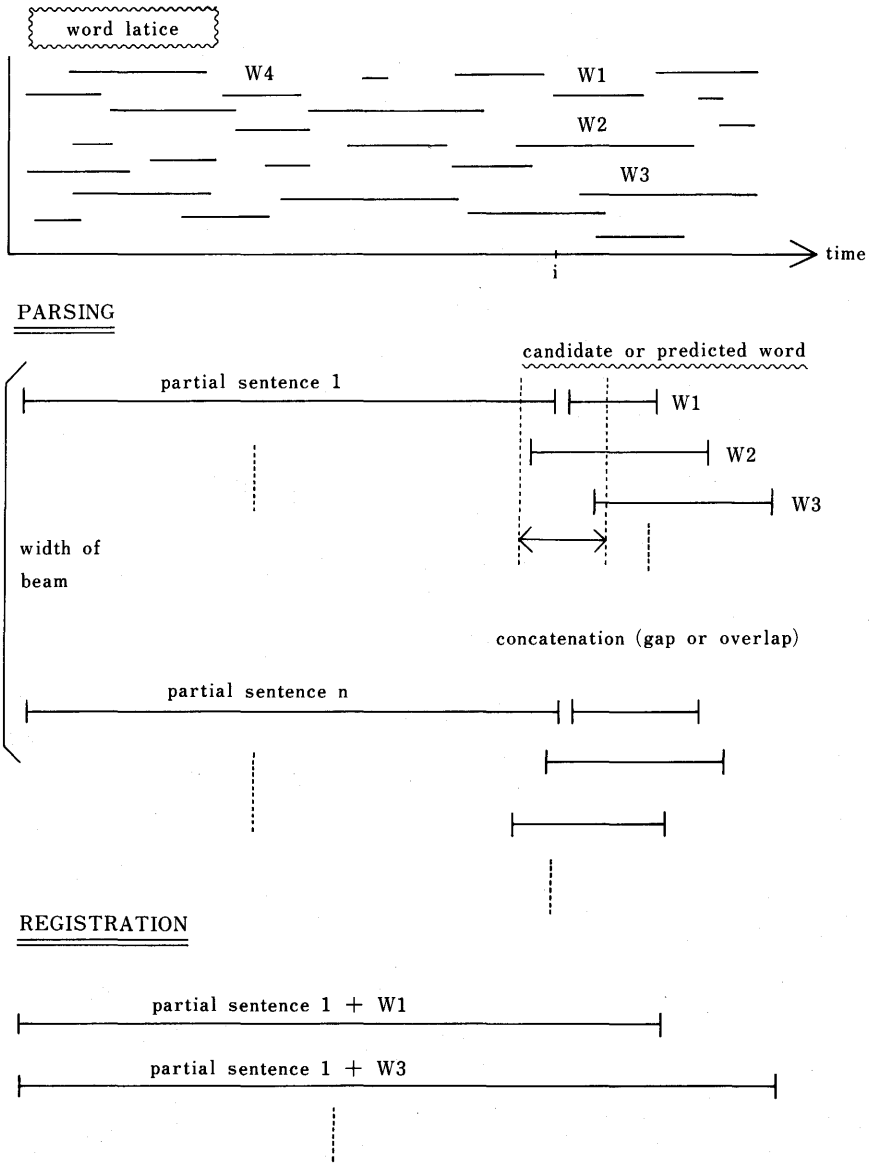

Fig. 7. A time-synchronous left-to-right parsing method (Forward Parsing Algorithm S*)

4. PREDICT(i, r) ← {predicted word} by PARSER(GPATH(i, r))
5. for all m such that CAND(m) ∈ {predicted words} by PREDICT(i, r), BEG(m) $>i-$gap2 and BEG(m) $<i+$gap1
   $t=$END(m)
   BREADTH(t) ← BREADTH(t)+1
   GPATH(t, BREADTH(t)) ← {grammar path} by PARSER(GPATH(i, r)) which predicts CAND(m)
   WSEQ(t, BREADTH(t))=WSEQ(i, r)⋈CAND(m)
   CSCORE(t, BREADTH(t))=CSCORE(i, r)+SCORE(m)×($t-i$)

If PARSER(GPATH(i, r)) generates plural grammar paths which predicts CAND(m), memorize all the plural GPATH, WSEQ and CSCORE. Note that the contents of WSEQ and CSCORE for all the plural memories are the same, respectively.

Sort CSCORE(t, k) for $k=1, 2, \cdots,$ BREADTH(t) and select only the top BEAM partial sentences, cumulative scores, and grammar paths on the basis of sorted cumulative scores. If BREADTH(t)>BEAM, BREADTH(t) $\leftarrow$ BEAM.

5. Island-Driven & Bottom-up Strategy

In an island-driven parsing strategy, partial hypotheses about the possible identity of utterance are generated from initial "seed" words in the utterance and are grown into larger and larger "island" hypotheses by the addition of words to the right/left hand side of the island.

In the HWIM system[1], the prediction of possible words at the right/left hand side of the island is performed by using the Augmented Transition Network grammar. We modified this prediction process to a standard context-free grammar[12]. The management of history of production rules used for the derivation of a partial sentence and prediction mechanism are the same as the left-to-right parsing mechanism. Beside such a mechanism, it is necessary to connect the two adjacent islands[13].

The parsing algorithm is given below.

(1) At the first step, the reliable "seeds" corresponding to the width of beam search are detected from the word lattice, and set $i=1$.

(2) The seeds or islands, the length of which is i, are expanded. If two islands are able to connect each other, these islands are integrated to an island.

(3) The number of islands with the same length is limited by the preset value, that is, the width or radis of beam search. Only the some islands with higher scores are selected and stored.

(4) If the sentence with the highest score which covers the whole of the utterance is hypothesized, the sentence is decided as the recognition result. Otherwise, $i \leftarrow i+1$ and go to step (2).

Fig. 8 illustrates this procedures.

6. Parsing Strategy for Function Word, Missing Word and Silence

It may occur that a function word or a short word is omitted by a word spotting algorithm. We should take the omission of function words into consideration, because the function word has very important role on the syntactical analysis. Therefore, we modify the above algorithm such that the parsing pro-

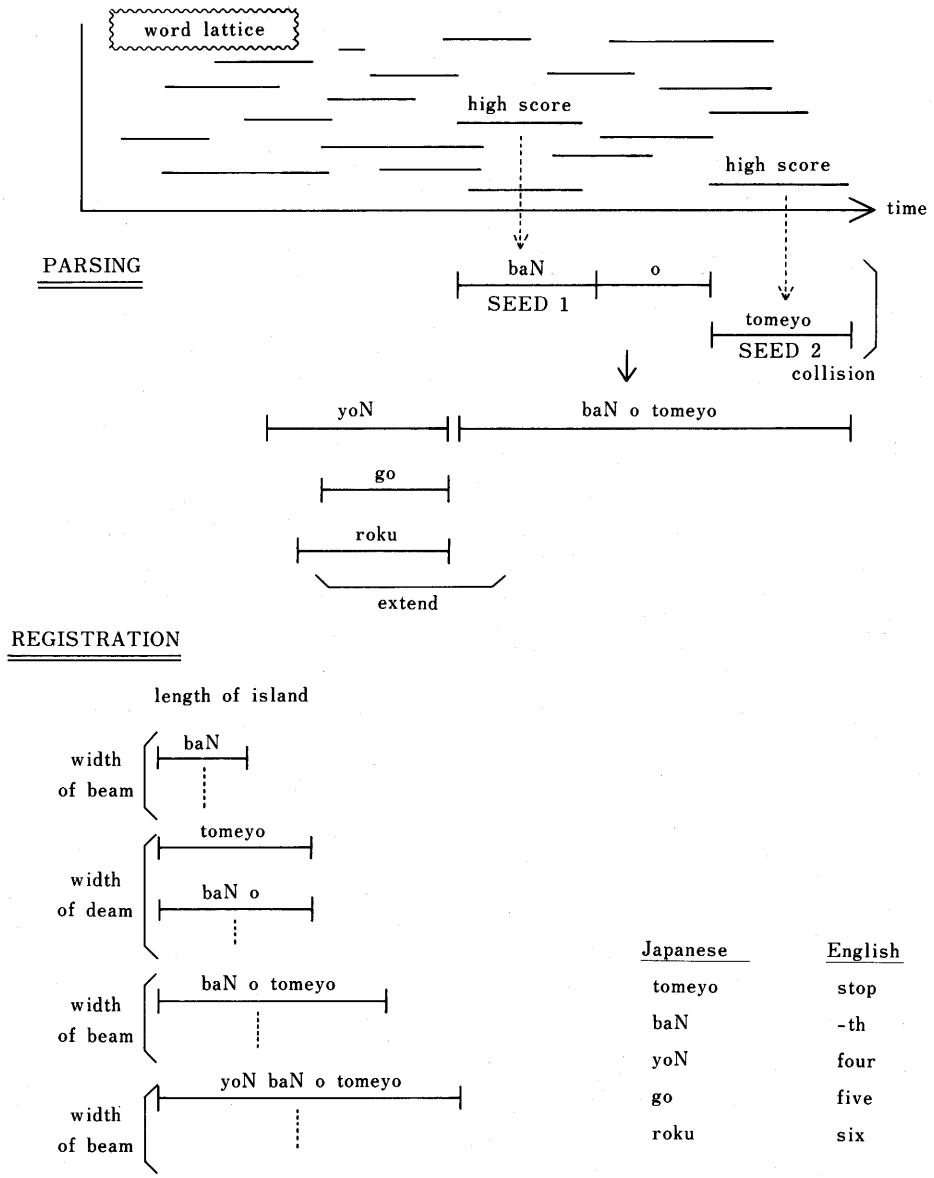

Fig. 8. An island-driven parsing method

Input : zikiteepu sooti yoN baN o tomeyo (Stop the four-th magnetic tape device)

ceeds even if a function word is missed. The modification is the following : We assume that a function word is spotted at an arbitrary locations by a reasonable score and length. The matching score is a default value. This assumption is put in step 5 of the left-to-right parsing algorithm.

We should also take the silences into consideration, because a silence may appear between words. One solution is to regard a silence pattern as a word pattern. There is no syntactical constraint for the silence, because the silences appear between arbitrary words. The other solution is to pay special attention to

a silence. We adopted the latter approach and implemented it as follows: If there exists a silence part in the beginning region of a spotted word, i. e., in the BEG(m) $-$gap1$\sim$BEG(m)$+$gap2, we widen the region to "the beginning frame of the silence" $-$gap1$\sim$BEG(m)$+$gap2. This operation is equal to delete the silences from the lattice. However, the silence gives the important information to the system, because there exists a choked sound in Japanese.

7. Comparison Results

7. 1 Task definition and word lattice

We used the three tasks of "computer network", "department store guidance", and "UNIX Q-A" as the evaluation task, which had adopted in the LITHAN[2] and SPOJUS-SYNO[14] speech understanding systems. The vocabulary size was about one, two, or five hundreds, respectively. Four types of task are simulated as follows:

(a) Small Task: "computer network" with the vocabulary size of 104 words and perplexity of test set=3.3.

(b) Medium Task: "computer network" with the vocabulary size of 250 words and perplexity=5.3.

(c) Combined Task: combination task of "computer network" and "department store guidance" with the total vocabulary size of 250 words and perplexity=3.4.

(d) Large Task: "UNIX Q-A" with the vocabulary size of 521 words and perplexity=10.0.

The phoneme recognizer was simulated by a random generater based on the

Table 1. Evaluation of Word Spotting.

(a) Phoneme Recognition Rate=60 %

Task	accuracy of spotted word (top n choices)					
	1	2	5	10	missing (total)	number of spotted word
small	55.4	69.1	86.3	91.8	6/489	857
medium	42.9	64.0	81.0	86.7	6/489	1477

(b) Phoneme Recognition Rate=80 %

Task	accuracy of spotted word (top n choices)					
	1	2	5	10	missing (total)	number of spotted word
small	80.8	88.3	95.9	97.5	0/489	931
medium	77.9	87.1	94.7	96.7	0/489	1605
combined	77.1	80.5	94.5	96.7	0/489	1745
large	53.5	66.7	78.6	85.5	4/387	3930

confusion matrix of a phoneme recongizer. We set the phoneme recognition rate to 60 % or 80 % and both the insertion and omission error rates to 5 %, respectively. The word lattice was generated by a word spotting algorithm from the generated phoneme sequence. The number of spotted words was about 400 ~ 5000 words per utterance.

Table 1 shows the evaluation results of the word spotting performance. The rate in the columm of the n-th choice shows the percent accuracy by which an input word is correctly identified as one among the best n spotted words in the neighborhood. The number of missing words denotes the total number of undetected input words in the total of 50 sentences (489 words for "computer network" and 387 words for "UNIX Q-A"). The word spotting performance of the large task was remarkably worse than that of the medium or combined task, because there were many confusable words in the task.

7. 2 Parsing results

(a) normal utterance

Table 2. Sentence Recognition Results for the Small Task.

PR : phoneme recognition rate
SR : sentence recognition rate
PW : predicted words/utterance
BF : average static branching factor
T : processing time/utterance (Apollo Domain 4000, C language)

Parsing Strategy	width of beam	PR=60 %				PR=80 %			
		SR %	PW	BF	T sec	SR %	PW	BF	T sec
left-to-right	5	50	391	5.7	6.4	66	505	6.4	7.7
	10	66	801	6.8	8.9	78	886	6.7	10.1
	20	66	1351	6.9	12.4	84	1430	5.9	14.0
island-driven	5	54	513	3.5	16.2	70	511	3.5	17.0
	10	66	1005	3.7	25.7	78	1063	3.6	27.7
	20	68	1935	3.7	50.1	84	2007	3.6	52.5

Table 3. Sentence Recognition Results for the Medium Task.

Parsing Strategy	width of beam	PR=60 %				PR=80 %			
		SR %	PW	BF	T sec	SR %	PW	BF	T sec
left-to-right	5	44	887	14.0	19.0	68	1249	16.7	22.5
	10	58	1870	17.3	23.7	72	2384	18.1	28.5
	20	60	3334	17.4	38.2	82	3933	15.0	38.2
island-driven	5	44	1042	7.3	37.5	68	1019	7.5	41.1
	10	58	1952	7.9	52.4	74	2109	8.2	59.5
	20	60	3876	8.2	90.3	82	4008	8.4	98.0

The parsing results by the two strategies are shown in Tables 2, 3, 4 and 5. The total number of simulated sentences was 50 sentences. The average number of words contained in a sentence was about 8 ~ 10 words. The average static

Table 4. Sentence Recognition Results for the Combined Task.

Parsing Strategy	width of beam	PR=80 %			
		SR %	PW	BF	T sec
left-to-right	5	66	649	5.6	22.4
	10	82	1042	5.4	25.8
	20	84	1612	5.3	31.4
island-driven	5	68	469	3.4	39.3
	10	82	967	3.5	53.0
	20	84	1741	3.4	80.9

Table 5. Experimental results for the vocabulary size of 521 words (large task).

Parsing Strategy	width of beam	PR=80 %			
		SR %	PW	BF	T sec
left-to-right	40	52	6566	22.3	349.2
island-driven	40	52	6829	27.4	659.8

branching factor denotes the average number of predicted words at time at the left or right hand side of a partial sentence. From the table, we can conclude in the following:

(1) The number of predicted words in direct proportional to the width of beam search, except 80 % of the phoneme recognition rate in the left-to-right parsing. In the case of 80 %, the total number of partial sentences at a frame was sometimes less than the width of beam search.

(2) The left-to-right & top-down parsing strategy was superior to the island-driven & bottom-up strategy in terms of the processing time.

(3) The recognition accuracy was almost the same for both strategies.

(4) The sentence recognition accuracy decreases according with the vocabulary size, but it is not so sensitive in the size.

(5) The sentence recognition accuracy is insensitive for the combined task, which consists of the different set of words. This fact would be well understood from no difference for the average static branching factor (BF) between the small task and the combined task.

(b) noisy utterance

Next, we simulated the recognition of spoken sentence, the head of which was disturbed by noises. In order to simulate it, we decreased the scores of first two words of an input utterance in the word lattice. We generated two kinds of noisy utterances.

Table 6 shows this simulation result. As expected, the island-driven parsing method was superior to the left-to-right passing method for a noisy utterance.

Table 6. Experimental results in the case of noisy words in the head part of input sentence (small task).

Parsing Strategy	width of beam	Phoneme Recognition Rate=80 %							
		SCORE←SCORE—50				SCORE←SCORE—25			
		SR %	PW	BF	T sec	SR %	PW	BF	T sec
left-to-right	5	26	502	6.3	7.6	62	505	6.4	7.6
	10	26	876	6.6	10.0	72	883	6.7	10.0
	20	26	1431	5.8	14.0	78	1429	5.9	14.0
island-driven	5	64	561	3.5	17.2	68	627	3.8	18.6
	10	66	1107	3.6	27.7	78	1242	4.0	30.6
	20	70	2032	3.7	52.0	82	2354	4.1	59.6

Table 7. Experimental results of spoken sentence recognition in the case of missing postposition (small task).

Parsing Strategy	width of beam	PR=60 %				PR=80 %			
		SR %	PW	BF	T sec	SR %	PW	BF	T sec
left-to-right	5	22	277	5.0	6.1	40	404	4.9	7.5
	10	34	744	5.9	9.3	50	786	5.7	10.0
	20	64	1263	6.7	13.1	80	1278	5.7	14.3
island-driven	5	62	535	4.1	15.9	76	617	4.2	17.5
	10	64	1128	4.2	26.8	80	1183	4.4	28.0
	20	64	2315	4.3	53.9	84	2355	4.3	55.9

(c) missing of functional words

Finally, we simulated such cases those a system missed to detect functional words (postpositions). So, we used the parsing mechanism as described in chapter 6. The score of postpositions in arbitrary locations was set at the default value. The result is shown in Table 7. In this table, we can conclude that our parsing mechanism for missing words worked well, and the computation did not increase, nevertheless the number of words in the lattice essentially increased.

8. Conclusion

We compared the two parsing mechanisms for continuous speech recognition. One was the left-to-right & top-down parser and the other was the island-driven & bottom-up parser.

From simulation results, we found that the left-to-right & top-down parser was superior to the island-driven & bottom-up parser on the recognition rate and parsing time (or predicted words). And also we found that the sentence recognition rate was not so sensitive on the vocabulary size, or on the combined task in comparison with a single task. Strictly speaking, it depends on both the entropy of a given language grammar (task) and the perplexity of phoneme (or word)[15].

However, when the initial part of an utterance was noisy, the island-driven strategy became superior to the left-to-right strategy. The further reseach is necessary to decide which is better. Finally, we found that our proposed parsing mechanism for missing function words was effective.

REFERENCES

1) W. A. Woods: "Optimal Search Strategies for Speech Understanding Control", Artificial Intelligence, Vol. 18, pp. 295-326 (1982).
2) T. Sakai and S. Nakagawa: "Speech Understanding System of Simple Japanese Sentences in a Task Domain", Trans. IECE, Vol. 60E, No. 1, pp. 13-20 (1977).
3) B. T. Lowerre: "Harpy Speech Recognition System", PhD thesis Carnegie Mellon University (1976).
4) N. J. Nilsson: "Problem-Solving Methods in Artificial Intelligence", McGraw-Hill (1971).
5) S. E. Levinson: "The Effects of Syntactic Analysis on Word Recognition Accuracy", AT & T Bell Lab. Tech. Jour., Vol. 57, No. 5, pp. 1627-1644 (1978).
6) L. G. Miller and S. E. Levinson: "Syntactic analysis for Large Vocabulary Speech Recognition Using a Contex-Free Covering Grammar", Proc. Int. Conf. Acoust., Speech & Signal Process, pp. 271-274 (1988).
7) W. H. Ward et at.: "Parsing Spoken Phrases Despite Missing Words", Proc. Int. Conf. Acoust., Speech & Signal Processing, pp. 275-278 (1988).
8) C. S. Christiansen and C. K. Rushforth: "Detecting and Locating Keywords in Continuous Speech Using Linear Predictive Coding", IEEE Trans. Acoust., Speech & Signal Processing, ASSP-25, pp. 361-367 (1977).
9) J. Earley: "An Efficient Context-Free Parsing Algorithms", Comm. ACM. 13, 2, pp. 94-102 (1970).
10) S. Nakagawa: "Speaker-Independent Sentence Recognition by Phoneme-Based Word Spotting and Context-Free Parsing", Carnegie Mellon University, Technical Report, CMU-CS 86-109 (1986).
11) S. Nakagawa: "Connected Spoken Word Recognition Algorithm by Constant Time Delay DP, O(n)DP and Augmented Continuous DP Matching", Information Science, Vol. 33, pp. 63-85 (1984).
12) Y. Ohguro and S. Nakagdwa: "Some Considerations on Parsing Methods by Left-to-Right & Top-Down and Island-Driven & Bottom-Up for Continuous Speech Recognition", IEICE Technical Report, NLC87-12 (1987-10, in Japanese).
13) Y. Kobayashi Y. Niimi and S. Uzuhara: "Linguistic processing in an Island-Driven Speech Understanding System", Trans. IEICE, Vol. 71-D, No. 12, pp. 2563-2570 (1988, in Japanese).
14) S. Nakagawa, Y. Ohguro and Y. Hashimoto: "Syntax-Oriented Speech Understanding System-SPOJUS-SYNO", Proc 2nd Int. Sym. on Advaneed Man-Machine Interface Through Spoken Language", pp. 32.1-32.12 (1988).
15) Y. Ohguro and S. Nakagawa: "Mutual Relationship Among Phoneme Recognition Accuracy, Perplexity and Sentence Recognition Accuracy", Report of Tech. IEICE, SP88-113 (1988, in Japanese)

(Received Oct. 31, 1988)