TR-I-0161

Generalized LR Parsing in Hidden Markov Model

Kenji Kita

ATR Interpreting Telephony Research Laboratories

1990.4

Abstract

This report describes a continuous speech recognition method called HMM-LR. This method uses an efficient parsing mechanism, a generalized LR parser, driving an HMM-based speech recognizer directly without any intervening structures such as a phoneme lattice. Very accurate, efficient speech recognition/parsing is achieved through the integrated processes of speech recognition and language analysis.

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

Generalized LR Parsing in Hidden Markov Model

1 Introduction

This report describes the application of Generalized LR parsing [21,22] to speech recognition. In particular, we will focus on a method called *HMM-LR*, first introduced by [8], which is an integration of Hidden Markov Models [12,18,19] and Generalized LR parsing.

Speech recognition is a technology that transforms an utterance into a text. With this technology, we can construct an intelligent machine that can listen to a speech utterance and then carry out the instructions appropriately. There have been many approaches to speech recognition, for example:

1. The *Feature-Based* approach [6,26],
2. The *Neural Network* approach [14,23],
3. The *Hidden Markov Model* approach [5,11].

Among these, the Hidden Markov Model is now the most widely used approach. The Hidden Markov Model is a powerful stochastic model and it has the ability to cope with the acoustical variations of speech. Moreover, any word models can be composed of phone models, thus it is easy to construct a large vocabulary speech recognition system.

One of the major problems in speech recognition is coping with large search spaces. As search space size increases, recognition performance decreases. Grammatical constraints are effective in reducing the search space and hence increase processing speed and recognition accuracy. Generalized LR parsing is one of the best mechanisms for dealing with grammatical constraints based on a context-free grammar.

There are two approaches using natural language parsers. In the first approach, the speech recognizer and the parser are used independently, that is, the speech recognizer is used to produce a phone/word *lattice* (a set of hypothesized phones/words with different starting and ending positions in the input speech). Then the parser is applied to obtain legal phone/word sequences. The second approach is to use grammatical constraints during speech recognition. It goes without saying that the first approach is not desirable, because of the information loss due to signal-symbol conversion. In HMM-LR, the LR parser drives Hidden Markov Models directly without any intervening structures such as a phone/word lattice. Thus, very accurate and efficient speech parsing is achieved.

2 Generalized LR Parsing

2.1 LR Parsing

LR parsing [1] was originally developed for programming languages. It is applicable for a large class of context-free grammars.

The LR parser is deterministically guided by an LR parsing table with two subtables (*action table* and *goto table*). The action table determines the next parser action $ACTION[s, a]$ from the state s currently on top of the stack and the current input symbol a. There are four kinds of actions, *shift*, *reduce*, *accept* and *error*. *Shift* means shift one word from input buffer onto the stack, *reduce* means reduce constituents on the stack using the grammar rule, *accept* means input is accepted by the grammar, and *error* means input is not accepted by the grammar. The goto table determines the next parser state $GOTO[s, A]$ from the state s and the grammar symbol A.

The LR parsing algorithm is summarized below.

1. *Initialization.* Set p to point to the first symbol of the input. Push the initial state 0 on top of the stack.

2. Consult $ACTION[s, a]$ where s is the state on top of the stack and a is the symbol pointed to by p.

3. If $ACTION[s, a] =$"*shift* s'", push s' on top of the stack and advance p to the next input symbol.

4. If $ACTION[s, a] =$"*reduce* $A \rightarrow \beta$", pop $|\beta|$ symbols off the stack and push $GOTO[s', A]$ where s' is the state now on top of the stack.

5. If $ACTION[s, a] =$"*accept*", parsing is completed.

6. If $ACTION[s, a] =$"*error*", parsing fails.

7. Return to 2.

2.2 Generalized LR Parsing

Standard LR parsing cannot handle ambiguous grammars. For an ambiguous grammar, the LR parsing table will have *multiple entries* (conflicts). As a general method, a stack-splitting mechanism can be used to cope with multiple entries. Whenever a multiple entry is encountered, the stack is divided into two stacks, and each stack is processed in parallel. Thus, it is possible to use LR parsing to handle an ambiguous grammar which describes natural language.

A simple example grammar is shown in Table 1, and the LR parsing table, compiled from the grammar automatically, is shown in Table 2. The left part is the action table and the right part is the goto table. The entry "*acc*" stands for the action "accept", and blank spaces represent "error". The terminal symbol "$" represents the end of the input.

3 Hidden Markov Models

Hidden Markov Models (HMM) are effective in expressing speech statistically and have been used widely for speech recognition.

3.1 Basic Concepts

An example of a Hidden Markov Model is shown in Figure 3. A model has a collection of *states* connected by *transitions*. Two sets of probabilities are attached to each transition. One is a *transition probability* a_{ij}, which provides the probability for taking a transition from state i to state j. The other is an *output probability* $b_{ij}(k)$, which provides the probability of emitting symbol k when taking a transition from state i to state j.

Formally, a Hidden Markov Model M is defined by a 4-tuple $M = (S, Y, A, B)$.

- S : A set of states $\{s_i\}$ including an initial state S_I and a final state S_F.

- Y : A set of output symbols.

- A : A set of transitions $\{a_{ij}\}$ where a_{ij} is the probability of taking a transition from state i to state j, and $\sum_j a_{ij} = 1$.

- B : The output probability distribution $\{b_{ij}(k)\}$ where $b_{ij}(k)$ is the probability of emitting symbol k when taking a transition from state i to state j, and $\sum_k b_{ij}(k) = 1$.

Typically, *vector quantization* (VQ) [13] is used as the acoustic frond-end for HMM. Vector quantization is a discrete representation of spectral space. A set of fixed prototype vectors is called a *codebook*, and an output symbol of a model comes from these prototype vectors.

(1)	S	$\rightarrow$	$NP\ V$
(2)	S	$\rightarrow$	V
(3)	NP	$\rightarrow$	N
(4)	NP	$\rightarrow$	$N\ P$
(5)	N	$\rightarrow$	$k\ o\ r\ e$
(6)	P	$\rightarrow$	o
(7)	V	$\rightarrow$	$k\ u\ r\ e$
(8)	V	$\rightarrow$	$o\ k\ u\ r\ e$

Figure 1: An example of a grammar

	e	o	u	k	r	$\$$	S	NP	V	P	N
0		$s3$		$s4$			1	5	2		6
1						acc					
2						$r2$					
3				$s7$							
4		$s8$	$s9$								
5		$s3$		$s11$					10		
6		$s13, r3$		$r3$						12	
7			$s14$								
8					$s15$						
9					$s16$						
10						$r1$					
11			$s9$								
12		$r4$		$r4$							
13		$r6$		$r6$							
14					$s17$						
15	$s18$										
16	$s19$										
17	$s20$										
18		$r5$		$r5$							
19						$r7$					
20						$r8$					

Figure 2: An example of an LR parsing table

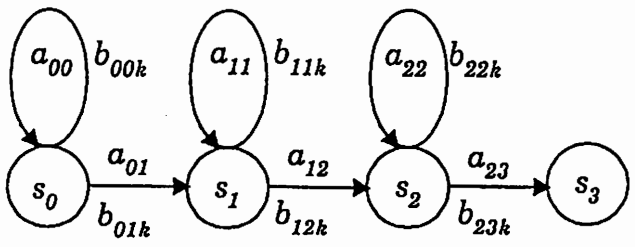

Figure 3: An example of a Hidden Markov Model

3.2 Recognition Problem

In a stochastic approach, having observed acoustic data y, a speech recognizer must decide a word sequence $\hat{w}$ that satisfies the following condition:

$$P(\hat{w}|y) = \max_{w} P(w|y)$$

By Bayes' rule,

$$P(w|y) = \frac{P(y|w)P(w)}{P(y)}$$

Since $P(y)$ does not depend on w, maximizing $P(w|y)$ is equivalent to maximizing $P(y|w)P(w)$. $P(w)$ is the *a priori* probability that the word sequence w will be uttered, and is estimated by the *language model.* In the case of a language model where all words are equally likely, this term is negligible. $P(y|w)$ is estimated by the *acoustic model.* Here we are using HMM as an acoustic model. Next we address the problem of how to estimate $P(y|w)$.

The *forward algorithm* can be used to compute the probability that a given model generated an observation sequence.

$$\alpha_i(t) = \begin{cases} 0 & t = 0 \;\&\; i \neq S_I \\ 1 & t = 0 \;\&\; i = S_I \\ \sum_j \alpha_j(t-1)a_{ji}b_{ji}(y_t) & t > 0 \end{cases}$$

$\alpha_i(t)$ is the probability that the Markov process is in state i having generated the sequence $y_1, y_2, \ldots, y_T$. The final probability for the model is given by $\alpha_{S_F}(T)$.

The trellis diagram in Figure 4 shows all possible state sequences. Each circle includes the cumulative probability at a particular state and time.

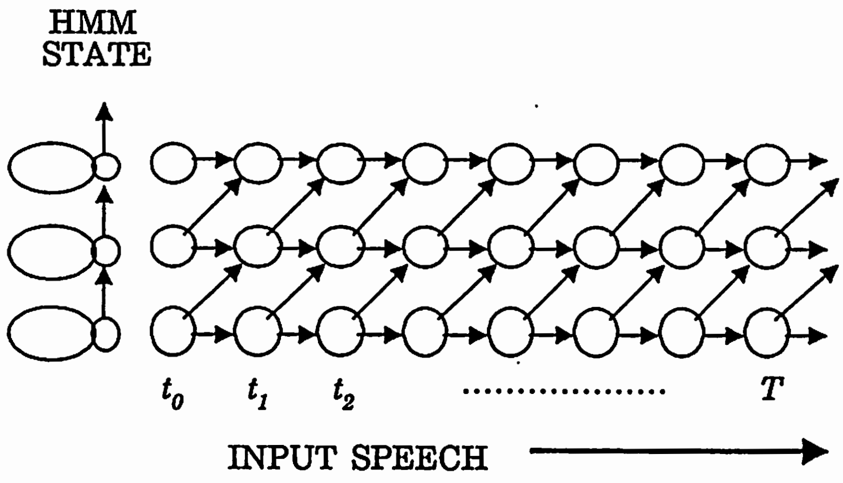

Figure 4: A trellis diagram

A slightly modified version of the forward algorithm is known as the *Viterbi algorithm.*

$$\alpha_i(t) = \begin{cases} 0 & t = 0 \;\&\; i \neq S_I \\ 1 & t = 0 \;\&\; i = S_I \\ \max\limits_j \alpha_j(t-1) a_{ji} b_{ji}(y_t) & t > 0 \end{cases}$$

In each recursion, if we remember the state that has the highest probability, we can obtain the most likely state sequence in the model that produced the observations.

3.3 Estimation of HMM Parameters

The parameters of the model (transition probabilities and output probabilities) can be estimated by the *forward-backward algorithm* given a collection of training data. The forward-backward algorithm is also known as the *Baum-Welch algorithm*. It is based on the *Maximum Likelihood Estimation.*

First, we define the *backward calculation* $\beta_i(t)$, which is a counterpart of the forward calculation $\alpha_i(t)$.

$$\beta_i(t) = \begin{cases} 0 & t = T \;\&\; i \neq S_F \\ 1 & t = T \;\&\; i = S_F \\ \sum\limits_j a_{ij} b_{ij}(y_{t+1}) \beta_j(t+1) & 0 \leq t < T \end{cases}$$

$\beta_i(t)$ is the probability that the Markov process is in state i, and will generate the sequence $y_{t+1}, y_{t+2}, \ldots, y_T$.

Now given some initial parameters of the model, we could re-estimate them using the following iterative calculations.

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T} \alpha_i(t-1) a_{ij} b_{ij}(y_t) \beta_j(t)}{\sum_{t=1}^{T} \sum_{k} \alpha_i(t-1) a_{ik} b_{ik}(y_t) \beta_k(t)}$$

$$\bar{b}_{ij}(k) = \frac{\sum_{t:y_t=k} \alpha_i(t-1) a_{ij} b_{ij}(y_t) \beta_j(t)}{\sum_{t=1}^{T} \alpha_i(t-1) a_{ij} b_{ij}(y_t) \beta_j(t)}$$

The forward-backward algorithm has been proven to converge [2].

4 HMM-LR Method

4.1 Basic Mechanism

This subsection gives an informal description of the HMM-LR method. We assume here that the HMM unit is the phone, although HMM can be used to represent any unit of speech, for example, word-based HMM, syllable-based HMM and phone-based HMM, etc.

In standard LR parsing, the next parser action (*shift*, *reduce*, *accept* or *error*) is determined using the current parser state and next input symbol to check the LR parsing table. This parsing mechanism is valid only for symbolic data and cannot be applied simply to continuous data such as speech.

In HMM-LR, the LR parser is used as a language source model for word/phone prediction/generation. Thus, we will hereafter call the LR parser the *predictive LR parser*. A phone-based predictive LR parser predicts next phones at each generation step and generates many possible sentences as phone sequences. The predictive LR parser determines next phones using the LR parsing table of the specified grammar and splits the parsing stack not only for grammatical ambiguity but also for phone variation. Because the predictive LR parser uses context-free rules whose terminal symbols are phone names, the phonetic lexicon for the specified task is embedded in the grammar. An example of context-free grammar rules with a phonetic lexicon is shown in Figure 5. Rule (5) indicates the definite article "the" before consonants, while rule (6) indicates the "the"

(1)	*S*	→	*NP VP*
(2)	*NP*	→	*DET N*
(3)	*VP*	→	*V*
(4)	*VP*	→	*V NP*
(5)	*DET*	→	*/z/ /a/*
(6)	*DET*	→	*/z/ /i/*
(7)	*N*	→	*/m/ /ae/ /n/*
(8)	*N*	→	*/ae/ /p/ /a/ /l/*
(9)	*V*	→	*/iy/ /ts/*
(10)	*V*	→	*/s/ /ih/ /ng/ /s/*

Figure 5: An example of a grammar with phonetic lexicon

before vowels. Rules (7), (8), (9) and (10) indicate the words "man", "apple", "eats" and "sings", respectively.

The HMM-LR continuous speech recognition system (see Figure 6) consists of the predictive LR parser and HMM phone verifiers. First, the parser picks up all phones predicted by the initial state of the LR parsing table and invokes the HMM models to verify the existence of these predicted phones. The parser then proceeds to the next state in the LR parsing table. During this process, all possible partial parses are constructed in parallel. The HMM phone verifier receives a *probability array* (see Figure 7) which includes end point candidates and their probabilities, and updates it using an HMM probability calculation. This probability array is attached to each partial parse. When the highest probability in the array is under a certain threshold level, the partial parse is pruned. The parsing process proceeds in this way, and stops if the parser detects an accept action in the LR parsing table. In this case, if the best probability point reaches the end of the speech data, parsing ends successfully.

Very accurate, efficient speech parsing is achieved through the integrated processes of speech recognition and language analysis.

4.2 Algorithm

This subsection gives the algorithm for HMM-LR as a recognizer, which produces no parse trees. It is, however, easy to extend the algorithm to produce parse trees.

First, we introduce a data structure called a *cell.* A cell is a structure with information about one recognition candidate. The following are kept in the cell.

- *LR parsing stack*, with information for parsing control.
- *Probability array*, which includes end point candidates and their probabilities.

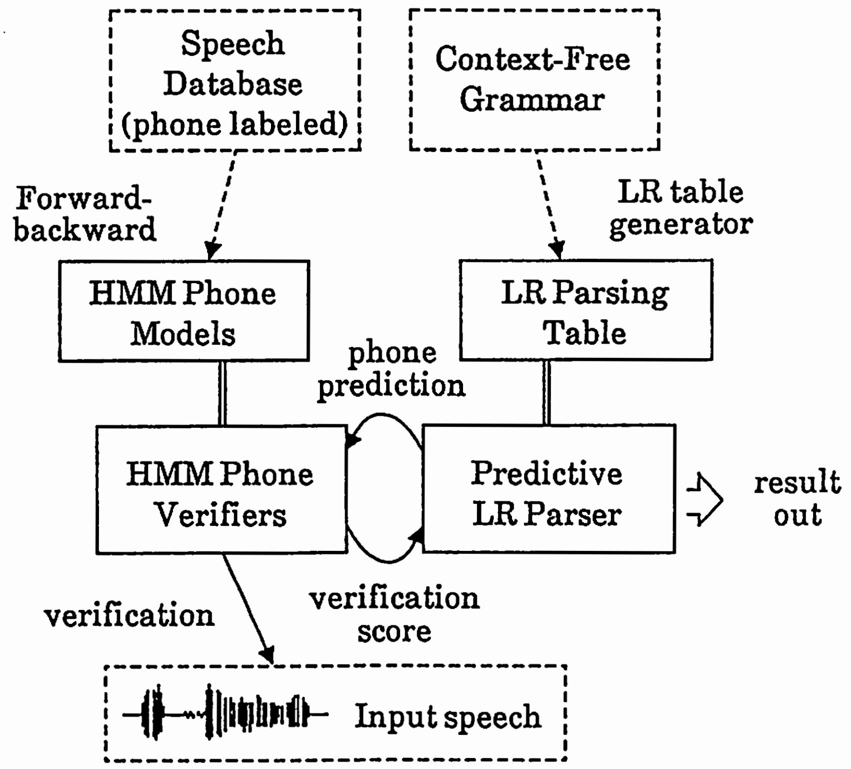

Figure 6: Schematic diagram of HMM-LR speech recognizer

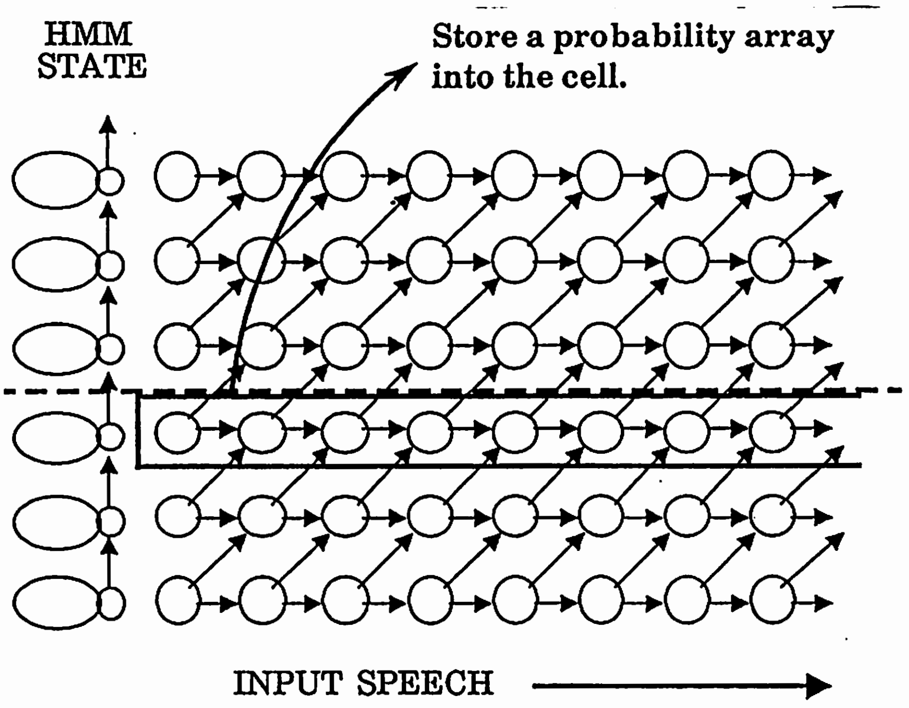

Figure 7: Stacking of a probability array

The algorithm is summarized below.

1. *Initialization.* Create a new cell C. Push the LR initial state 0 on top of the LR parsing stack of C. Initialize the probability array Q of C;

$$Q(t) = \begin{cases} 1 & t = 0 \\ 0 & 1 \leq t \leq T \end{cases}$$

2. *Ramification of cells.* Construct a set

$$\begin{aligned} S \;=\; & \{(C, s, a, x) | \exists C, s, a, x (C \text{ is a cell which is not accepted} \\ & \& \; s \text{ is the state on top of the LR parsing stack of } C \\ & \& \; x = ACTION[s, a] \; \& \; x \neq \text{“}error\text{”}\}) \end{aligned}$$

For each element $(C, s, a, x) \in S$, do operations below. If a set S is empty, parsing is completed.

3. If $x =$"*shift s'*", verify the existence of phone a. In this case, update the probability array Q of the cell C using the following computation.

$$\alpha_i(t) = \begin{cases} Q(t) & i = S_I \\ 0 & i \neq S_I \,\&\, t = 0 \\ \sum_j \alpha_j(t-1) a_{ji} b_{ji}(y_t) & i \neq 0 \,\&\, t > 0 \end{cases}$$

$$Q(t) = \begin{cases} 0 & t = 0 \\ \alpha_{S_F}(t) & t > 0 \end{cases}$$

   If $\max_{1 \leq t \leq T} Q(t)$ is below a certain threshold, cell C is abandoned. Else push s' on top of the LR parsing stack of cell C.

4. If $x =$ "*reduce* $A \rightarrow \beta$", pop $|\beta|$ symbols off the LR parsing stack and push $GOTO[s', A]$ where s' is the current state on top of the stack.

5. If $x =$ "*accept*" and $Q(T)$ exceeds a certain threshold, cell C is accepted. If not, cell C is abandoned.

6. Return to 2.

Recognition results are kept in accepted cells. Generally, many recognition candidates exist, and it is possible to rank these candidates using a value $Q(T)$ of each cell.

4.3 Refinements of the Algorithm

The algorithm described above is a simple one. It is possible to make some refinements to the algorithm.

1. Using the beam-search technique.
   The *beam-search* technique was first used in the HARPY speech recognition system [15]. It is a modification of the breadth-first search technique, in which a group of near-miss alternatives around the best path are selected and processed in parallel. The beam-search technique reduces search cost and maintains search efficiency. Generally, a set S constructed in step 2 in the algorithm is quite large. The beam-search technique can be used to select a group of likely cells. The value $\max_{1 \leq t \leq T} Q(t)$ of each cell can be used as an evaluation score.

2. Using the graph-structured stack.
   The *graph-structured stack* is one of the key ideas in Generalized LR parsing. In the above algorithm, when making a set S, copies of an LR parsing

stack are created. By using the graph-structured stack, it is not necessary to copy the whole stack. Copying only the necessary portion of the stack is sufficient. Thus, the amount of computation is reduced.

5 ATR HMM-LR Continuous Speech Recognition System

In this section, we describe the implementation of a continuous speech recognition system based on HMM-LR method, developed at *ATR Interpreting Telephony Research Laboratories* [5]. This system recognizes Japanese phrase-wise utterances, and is used as the frond-end of the *SL-TRANS*, a spoken language translation system from Japanese into English [17].

5.1 Signal Processing

The speech is sampled at 12 KHz, pre-emphasized with a filter whose transform function is $(1 - 0.97z^{-1})$, and windowed using a 256-point Hamming window every 9 msec. Then, 12-order LPC analysis is carried out. Spectrum, difference cepstrum coefficients, and power are computed. Multiple VQ codebooks for each feature were generated using 216 phonetically balanced words.

5.2 HMM Phone Models

A three-loop model for consonants and a one-loop model for vowels are trained using each phone data extracted from the ATR isolated word database [10].

To represent phone models with less distortion, separate vector quantization (multiple codebooks) are used, where spectrum, LPC cepstral difference and power are quantized separately. In the training stage the output vector probabilities of these three codebooks are estimated simultaneously and independently, and in the recognition stage all the output probabilities are calculated as the product of the output vector probabilities in these codebooks.

HMM is effective in expressing speech data statistically, but phone duration information from speech data is not modeled statistically in the HMM phone models. In order to make a statistical duration model, an HMM state duration control is realized as a state duration penalty calculated from an HMM state duration distribution of the training data.

As described above, HMM phone models were trained using the word utterances, whereas the recognition is carried out for continuous speech. To realize accurate duration control, HMM duration parameters were modified according to the speaking rates of word and phrase utterances.

5.3 Grammar

The grammar is designed to cover many linguistic phenomena common in Japanese. The complexity of the grammar is measured by ***task entropy*** and ***phone perplexity*** [7]. Task entropy is defined as average information obtained when an utterance is recognized correctly. The phone perplexity is defined as the average number of phones predicted at each step. The complexity of the grammar is summarized in Table 1.

There are 1,461 grammar rules including 1,035 different word, and the phone perplexity is 5.9. Assuming that the average phone length per word is three, the word perplexity will exceed 100.

Table 1: Grammar complexity

Vocabulary	1,035 words
Task Entropy	17.0
Phone Perplexity	5.9
Estimated Word Perplexity	more than 100

5.4 Performance

The system was tested for four speakers (three male, one female) both in the speaker-dependent condition and in the speaker-adapted condition.

The result is shown in Table 2. Average phrase recognition rate is 89.5%, and a rate of 99.3% is achieved for the top five choices in the speaker-dependent condition. In the speaker-adapted condition, rates are 80.2% and 98.1%, respectively.

Table 2: Recognition performance

Rank	Recognition Rate (%)	
	Speaker Dependent	Speaker Adapted
1	89.5	80.2
2	96.4	93.1
3	98.6	95.8
4	99.0	97.4
5	99.3	98.1

6 Concluding Remarks

In this report, we have described HMM-LR, an accurate and efficient speech recognition/parsing method. We have also introduced a speech recognizer based on this method.

In HMM-LR, Generalized LR parsing is used as a language source model for word/phoneme prediction/generation. This characteristic of Generalized LR parsing can be applied to other approaches of speech recognition. Indeed, Generalized LR parsing is successfully integrated with *Time-Delay Neural Networks* [23] and attains good performance [16].

We have not considered the stochastic language modeling so far. From the viewpoint of information theory, every language has its own information entropy which includes probabilities of word occurrences and probabilities of word sequences. For example, N-gram language models (bigram, trigram, etc.) are extensively used to correct recognition errors and improve recognition accuracy [11,20]. An N-gram language model is an extremely rough approximation of a language, but it is effective in correcting local syntax errors. Another approach to making a stochastic language model is to use a probabilistic context-free grammar [3,4] or a probabilistic LR parsing [24,25]. These stochastic language models can be incorporated into the HMM-LR speech recognizer, and attain better performance [9].

Acknowledgments

I would like to thank Dr. Akira Kurematsu, the president of ATR Interpreting Telephony Research Laboratories, for his encouragement and support. I would also like to thank Toshiyuki Hanazawa, Takeshi Kawabata, Kiyohiro Shikano, Hiroaki Saito and Tsuyoshi Morimoto for their valuable technical suggestions and comments.

References

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

[2] Baum, L. E., Petrie, T., Soules, G., Weiss, N., *A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains*, The Annals of Mathematical Statistics, Vol. 41, 1970.

[3] Fu, K. S., *Syntactic Methods in Pattern Recognition*, Academic Press, 1974.

[4] Fujisaki, T., Jelinek, F., Cocke, J., Black, E., Nishino, T., *A Probabilistic Parsing Method for Sentence Disambiguation*, International Workshop on Parsing Technologies, 1989.

[5] Hanazawa, T., Kita, K., Nakamura, S., Kawabata, T., Shikano, K., *ATR HMM-LR Continuous Speech Recognition System*, Proc. of ICASSP 90 - IEEE International Conference on Acoustics, Speech and Signal Processing, 1990.

[6] Hatazaki, K., Komori, Y., Kawabata, T., Shikano, K., *Phoneme Segmentation Using Spectrogram Reading Knowledge*, Proc. of ICASSP 89 - IEEE International Conference on Acoustics, Speech and Signal Processing, 1989.

[7] Kawabata, T., Shikano, K., Kita, K., *Task entropy and Phone Perplexity*, The Acoustic Society of Japan Spring Meeting Proc., 1989. (in Japanese).

[8] Kita, K., Kawabata, T., Saito, H., *HMM Continuous Speech Recognition Using Predictive LR Parsing*, Proc. of ICASSP 89 - IEEE International Conference on Acoustics, Speech and Signal Processing, 1989.

[9] Kita, K., Kawabata, T., Hanazawa, T., *HMM Continuous Speech Recognition Using Stochastic Language Models*, Proc. of ICASSP 90 - IEEE International Conference on Acoustics, Speech and Signal Processing, 1990.

[10] Kuwabara, H., Takeda, K., Sagisaka, Y., Katagiri, S., Morikawa, S., Watanabe, T., *Construction of a Large-Scale Japanese Speech Database and its Management System*, Proc. of ICASSP 89 - IEEE International Conference on Acoustics, Speech and Signal Processing, 1989.

[11] Lee, K. F., Hon, H. W., Reddy, R., *An Overview of the SPHINX Speech Recognition System*, IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. 38, No. 1, 1990.

[12] Levinson, S. E., Rabiner, L. R., Sondhi, M. M., *An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition*, Bell Syst. Tech. J., Vol. 62, No. 4, 1983.

[13] Linde, Y., Buzo, A., Gray, R. M., *An Algorithm for Vector Quantizer Design*, IEEE Transaction on Communications, COM-28, 1980.

[14] Lippmann, R. P., *Review of Neural Networks for Speech Recognition*, Neural Computation, Vol. 1, No. 1, 1989.

[15] Lowerre, B., Reddy, R., *The HARPY Speech Recognition System*, in Trends in Speech Recognition, ed. Lea, W. A., Prentice-Hall, 1980.

[16] Minami, Y., Sawai, H., Miyatake, M., Shikano, K., *Large Vocabulary Spoken Word Recognition Using Time-Delay Neural Network Phoneme Spotting and Predictive LR-Parsing*, Trans. Tech. Group Speech Acoust. Soc. Japan, SP89-99, 1990. (in Japanese).

[17] Morimoto, T., Ogura, K., Kita, K., Kogure, K., Kakigahara, K., *Spoken Language Processing in SL-TRANS*, ATR Symposium on Basic Research for Telephone Interpretation, 1989.

[18] Poritz, A. B., *Hidden Markov Models: A Guided Tour*, Proc. of ICASSP 88 - IEEE International Conference on Acoustics, Speech and Signal Processing, 1988.

[19] Rabiner, L. R., Juang, B. H., *An Introduction to Hidden Markov Models*, IEEE ASSP Magazine, 1986.

[20] Shikano, K., *Improvement of Word Recognition Results by Trigram Model*, Proc. of ICASSP 87 - IEEE International Conference on Acoustics, Speech and Signal Processing, 1987.

[21] Tomita, M., *Efficient Parsing for Natural Language - A Fast Algorithm for Practical Systems*, Kluwer Academic Publishers, 1986.

[22] Tomita, M., *An Efficient Augmented-Context-Free Parsing Algorithm*, Computational Linguistics, Vol. 13, No. 1-2, 1987.

[23] Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., Lang, K. J., *Phoneme Recognition Using Time-Delay Neural Networks*, IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. 37, No. 3, 1989.

[24] Wright, J. H., Wrigley, E. N., *Linguistic Control in Speech Recognition*, Proc. 7th FASE Symposium, 1988.

[25] Wright, J. H., Wrigley, E. N., *Probabilistic LR Parsing for Speech Recognition*, International Workshop on Parsing Technologies, 1989.

[26] Zue, V., Glass, J., Phillips, M., Seneff, S., *Acoustic Segmentation and Phonetic Classification in the SUMMIT System*, Proc. of ICASSP 89 - IEEE International Conference on Acoustics, Speech and Signal Processing, 1989.