FACTORIZATION OF LANGUAGE CONSTRAINTS IN SPEECH RECOGNITION

Roberto Pieraccini and Chin-Hui Lee
Speech Research Department
AT&T Bell Laboratories
Murray Hill, NJ 07974, USA

ABSTRACT

Integration of language constraints into a large vocabulary speech recognition system often leads to prohibitive complexity. We propose to factor the constraints into two components. The first is characterized by a covering grammar which is small and easily integrated into existing speech recognizers. The recognized string is then decoded by means of an efficient language post-processor in which the full set of constraints is imposed to correct possible errors introduced by the speech recognizer.

1. Introduction

In the past, speech recognition has mostly been applied to small domain tasks in which language constraints can be characterized by regular grammars. All the *knowledge sources* required to perform speech recognition and understanding, including acoustic, phonetic, lexical, syntactic and semantic levels of knowledge, are often encoded in an *integrated* manner using a *finite state network* (FSN) representation. Speech recognition is then performed by finding the most likely path through the FSN so that the *acoustic distance* between the input utterance and the recognized string decoded from the most likely path is minimized. Such a procedure is also known as *maximum likelihood decoding*, and such systems are referred to as *integrated* systems. Integrated systems can generally achieve high accuracy mainly due to the fact that the decisions are delayed until enough information, derived from the knowledge sources, is available to the decoder. For example, in an integrated system there is no explicit segmentation into phonetic units or words during the decoding process. All the segmentation hypotheses consistent with the introduced constraints are carried on until the final decision is made in order to maximize a global function. An example of an integrated system was HARPY (Lowerre, 1980) which integrated multiple levels of knowledge into a single FSN. This produced relatively high performance for the time, but at the cost of multiplying out constraints in a manner that expanded the grammar beyond reasonable bounds for even moderately complex domains, and may not scale up to more complex tasks. Other examples of integrated systems may be found in Baker (1975) and Levinson (1980).

On the other hand *modular* systems clearly separate the knowledge sources. Different from integrated systems, a modular system usually make an explicit use of the constraints at each level of knowledge for making hard decisions. For instance, in modular systems there is an explicit segmentation into phones during an early stage of the decoding, generally followed by lexical access, and by syntactic/semantic parsing. While a modular system, like for instance HWIM (Woods, 1976) or HEARSAY-II (Reddy, 1977) may be the only solution for extremely large tasks when the size of the vocabulary is on the order of 10,000 words or more (Levinson, 1988), it generally achieves lower performance than an integrated system in a restricted domain task (Levinson, 1989). The degradation in performance is mainly due to the way errors propagate through the system. It is widely agreed that it is dangerous to make a long series of hard decisions. The system cannot recover from an error at any point along the chain. One would want to avoid this chain-architecture and look for an architecture which would enable modules to compensate for each other. Integrated approaches have this compensation capability, but at the cost of multiplying the size of the grammar in such a way that the computation becomes prohibitive for the recognizer. A solution to the problem is to factorize the constraints so that the size of the grammar, used for maximum likelihood decoding, is kept within reasonable bounds without a loss in the performance. In this paper we propose an approach in which speech recognition is still performed in an integrated fashion using a *covering grammar* with a smaller FSN representation. The decoded string of words is used as input to a second module in which the complete set of task constraints is imposed to correct possible errors introduced by the speech recognition module.

2. Syntax Driven Continuous Speech Recognition

The general trend in large vocabulary continuous speech recognition research is that of building integrated systems (Huang, 1990; Murveit, 1990; Paul, 1990; Austin, 1990) in which all the relevant knowledge sources, namely acoustic, phonetic, lexical, syntactic, and semantic, are integrated into a unique representation. The speech signal, for the purpose of speech recognition, is represented by a sequence of acoustic patterns each consisting of a set of measurements taken on a small portion of signal (generally on the order of 10 msec). The speech recognition process is carried out by searching for the best path that interprets the sequence of acoustic patterns, within a network that represents, in its more detailed structure, all the possible sequences of acoustic configurations. The network, generally called a *decoding network*, is built in a hierarchical way. In current speech recognition systems, the syntactic structure of the sentence is represented generally by a regular grammar that is typically implemented as a finite state network (*syntactic FSN*). The arcs of the syntactic FSN represent vocabulary items, that are again represented by FSN's (*lexical FSN*), whose arcs are phonetic units. Finally every phonetic unit is again represented by an FSN (*phonetic FSN*). The nodes of the phonetic FSN, often referred to as *acoustic states*, incorporate particular acoustic models developed within a statistical framework known as hidden Markov model (HMM).[1] The model pertaining to an acoustic state allows computation of a likelihood score, which represents the goodness of acoustic match for the observation of a given acoustic patterns. The decoding network is obtained by representing the overall syntactic FSN in terms of acoustic states.

1. The reader is referred to Rabiner (1989) for a tutorial introduction of HMM.

Therefore the recognition problem can be stated as follows. Given a sequence of acoustic patterns, corresponding to an uttered sentence, find the sequence of acoustic states in the decoding network that gives the highest likelihood score when aligned with the input sequence of acoustic patterns. This problem can be solved efficiently and effectively using a dynamic programming search procedure. The resulting optimal path through the network gives the optimal sequence of acoustic states, which represents a sequence of phonetic units, and eventually the recognized string of words. Details about the speech recognition system we refer to in the paper can be found in Lee (1990/1). The complexity of such an algorithm consists of two factors. The first is the complexity arising from the computation of the likelihood scores for all the possible pairs of acoustic state and acoustic pattern. Given an utterance of fixed length the complexity is linear with the number of *distinct* acoustic states. Since a finite set of phonetic units is used to represent all the words of a language, the number of possible different acoustic states is limited by the number of distinct phonetic units. Therefore the complexity of the local likelihood computation factor does not depend either on the size of the vocabulary or on the complexity of the language. The second factor is the combinatorics or bookkeeping that is necessary for carrying out the dynamic programming optimization. Although the complexity of this factor strongly depends on the implementation of the search algorithm, it is generally true that the number of operations grows linearly with the number of arcs in the decoding network. As the overall number of arcs in the decoding network is a linear function of the number of arcs in the syntactic network, the complexity of the bookkeeping factor grows linearly with the number of arcs in the FSN representation of the grammar.

The syntactic FSN that represents a certain task language may be very large if both the size of the vocabulary and the number of syntactic constraints are large. Performing speech recognition with a very large syntactic FSN results in serious computational and memory problems. For example, in the DARPA resource management task (RMT) (Price, 1988) the vocabulary consists of 991 words and there are 990 different basic sentence structures (*sentence generation templates*, as explained later). The original structure of the language (RMT grammar), which is given as a non-deterministic finite state semantic grammar (Hendrix, 1978), contains 100,851 rules, 61,928 states and 247,269 arcs. A two step automatic optimization procedure (Brown, 1990) was used to compile (and minimize) the nondeterministic FSN into a deterministic FSN, resulting in a machine with 3,355 null arcs, 29,757 non-null arcs, and 5832 states. Even with compilation, the grammar is still too large for the speech recognizer to handle very easily. It could take up to an hour of cpu time for the recognizer to process a single 5 second sentence, running on a 300 Mflop Alliant supercomputer (more that 700 times slower than real time). However, if we use a simpler *covering* grammar, then recognition time is no longer prohibitive (about 20 times real time). Admittedly, performance does degrade somewhat, but it is still satisfactory (Lee, 1990/2) (e.g. a 5% word error rate). A simpler grammar, however, represents a superset of the domain language, and results in the recognition of word sequences that are outside the defined language. An example of a covering grammars for the RMT task is the so called *word-pair* (WP) grammar where, for each vocabulary word a list is given of all the words that may follow that word in a sentence. Another covering grammar is the so called null grammar (NG), in which a word can follow any other word. The average word branching factor is about 60 in the WP grammar. The constraints imposed by the WP grammar may be easily imposed in the decoding phase in a rather inexpensive procedural way, keeping the size of the FSN very small (10 nodes and 1016 arcs in our implementation (Lee, 1990/1) and allowing the recognizer to operate in a reasonable time (an average of 1 minute of CPU time per sentence) (Pieraccini, 1990). The sequence of words obtained with the speech recognition procedure using the WP or NG grammar is then used as input to a second stage that we call the *semantic decoder.*

3. Semantic Decoding

The RMT grammar is represented, according to a context free formalism, by a set of 990 *sentence generation templates* of the form:

$$S_j = a_1^j \; a_2^j \; \cdots \; a_{N_j}^j \qquad (1)$$

where a generic a_i^j may be either a terminal symbol, hence a word belonging to the 991 word vocabulary and identified by its orthographic transcription, or a non-terminal symbol (represented by sharp parentheses in the rest of the paper). Two examples of sentence generation templates and the corresponding production of non-terminal symbols are given in Table 1 in which the symbol ε corresponds to the empty string.

A characteristic of the the RMT grammar is that there are no recursive productions of the kind:

$$\langle A \rangle = a_1 \; a_2 \; \cdots \; \langle A \rangle \; \cdots \; a_N \qquad (2)$$

For the purpose of semantic decoding, each sentence template may then be represented as a FSN where the arcs correspond either to vocabulary words or to categories of vocabulary words. A category is assigned to a vocabulary word whenever that vocabulary word is a unique element in the right hand side of a production. The category is then identified with the symbol used to represent the non-terminal on the left hand side of the production. For instance, following the example of Table 1, the words SHIPS, FRIGATES, CRUISERS, CARRIERS, SUBMARINES, SUBS, and VESSELS belong to the category <SHIPS>, while the word LIST belongs to the category <LIST>. A special word, the null word, is included in the vocabulary and it is represented by the symbol ε.

Some of the non-terminal symbols in a given sentence generation template are essential for the representation of the meaning of the sentence, while others just represent equivalent syntactic variations with the same meaning. For instance,

GIVE A LIST OF <OPTALL> <OPTTHE> <SHIPS>	
<LIST> <OPTTHE> <THREATS>	
<OPTALL>	ALL ε
<OPTTHE>	THE ε
<SHIPS>	SHIPS FRIGATES CRUISERS CARRIERS SUBMARINES SUBS VESSELS
<LIST>	SHOW <OPTME> GIVE <OPTME> LIST GET <OPTME> FIND <OPTME> GIVE ME A LIST OF GET <OPTME> A LIST OF
<THREATS>	ALERTS THREATS
<OPTME>	ME ε

TABLE 1. Examples of sentence generation templates and semantic categories

the correct detection by the recognizer of the words uttered in place of the non-terminals <SHIPS> and <THREATS>, in the former examples, is essential for the execution of the correct action, while an error introduced at the level of the nonterminals <OPTALL>, <OPTTHE> and <LIST> does not change the meaning of the sentence, provided that the sentence generation template associated to the uttered sentence has been correctly identified. Therefore there are non-terminals associated with essential information for the execution of the action expressed by the sentence that we call *semantic variables*. An analysis of the 990 sentence generation templates allowed to define a set of 69 semantic variables.

The function of the semantic decoder is that of finding the sentence generation template that most likely produced the uttered sentence and give the correct values to its semantic variables. The sequence of words given by the recognizer, that is the input of the semantic decoder, may have errors like word substitutions, insertions or deletions. Hence the semantic decoder should be provided with an error correction mechanism. With this assumptions, the problem of semantic decoding may be solved by introducing a distance criterion between a string of words and a sentence template that reflects the nature of the possible word errors. We defined the distance between a string of words and a sentence generation templates as the minimum Levenshtein[2] distance between the string of words and all the string of words that can be generated by the sentence generation template. The Levenshtein distance can be easily computed using a dynamic programming procedure. Once the *best matching* template has been found, a traceback procedure is executed to recover the modified sequence of words.

3.1 Semantic Filter

After the alignment procedure described above, a semantic check may be performed on the words that correspond to the non-terminals

2. The Levenshtein distance (Levenshtein, 1966) between two strings is defined as the minimum number of editing operations (substitutions, deletions, and insertions) for transforming one string into the other.

associated with semantic variables in the selected template. If the results of the check is positive, namely the words assigned to the semantic variables belong to the possible values that those variables may have, we assume that the sentence has been correctly decoded, and the process stops. In the case of a negative response we can perform an additional acoustic or phonetic verification, using the available constraints, in order to find which production, among those related to the considered non-terminal, is the one that more likely produced the acoustic pattern. There are different ways of carrying out the verification. In the current implementation we performed a *phonetic* verification rather than an acoustic one. The recognized sentence (i.e. the sequence of words produced by the recognizer) is transcribed in terms of phonetic units according to the pronunciation dictionary used in speech decoding. The template selected during semantic decoding is also transformed into an FSN in terms of phonetic units. The transformation is obtained by expanding all the non-terminals into the corresponding vocabulary words and each word in terms of phonetic units. Finally a matching between the string of phones describing the recognized sentence and the phone-transcribed sentence template is performed to find the most probable sequence of words among those represented by the template itself (phonetic verification). Again, the matching is performed in order to minimize the Levenshtein distance. An example of this verification procedure is shown in Table 2.

The first line in the example of Table 2 shows the sentence that was actually uttered by the speaker. The second line shows the recognized sentence. The recognizer deleted the word WERE, substituted the word THERE for the word THE and the word EIGHT for the word DATE. The semantic decoder found that, among the 990 sentence generation templates, the one shown in the third line of Table 2 is the one that minimizes the criterion discussed in the previous section. There are three semantic variables in this template, namely <NUMBER>, <SHIPS> and <YEAR>. The backtracking procedure associated to them the words DATE, SUBMARINES, and EIGHTY TWO respectively. The semantic check gives a *false* response for the variable <NUMBER>. In fact there are no productions of the kind <NUMBER> := DATE. Hence the recognized string is translated into its phonetic representation. This representation is aligned with the phonetic representation of the template and gives the string shown in the last line of the table as the best interpretation.

3.2 Acoustic Verification

A more sophisticated system was also experimented allowing for acoustic verification after semantic postprocessing.

For some uttered sentences it may happen that more than one template shows the very same minimum Levenshtein distance from the recognized sentence. This is due to the simple metric that is used in computing the distance between a recognized string and a sentence template. For example, if the uttered sentence is:

WHEN WILL THE PERSONNEL CASUALTY REPORT FROM THE YORKTOWN BE RESOLVED

uttered	WERE THERE MORE THAN EIGHT SUBMARINES EMPLOYED IN EIGHTY TWO	
recognized	THE MORE THAN DATE SUBMARINES EMPLOYED END EIGHTY TWO	
template	WERE THERE MORE THAN <NUMBER> <SHIPS> EMPLOYED IN <YEAR>	
semantic variable	value	check
<NUMBER>	DATE	FALSE
<SHIPS>	SUBMARINES	TRUE
<YEAR>	EIGHTY TWO	TRUE
phonetic	dh ae t m ao r t ay l ae n d d ey t s ah b m ax r iy n z ix m p l oy d eh n d ey dx iy t w eh n iy	
corrected	WERE THERE MORE THAN EIGHT SUBMARINES EMPLOYED IN EIGHTY TWO	

TABLE 2. An example of semantic postprocessing

and the recognized sentence is:

WILL THE PERSONNEL CASUALTY REPORT THE YORKTOWN BE RESOLVED

there are two sentence templates that show a minimum Levenshtein distance of 2 (i.e. two words are deleted in both cases) from the recognized sentence, namely:

1) <WHEN+LL> <OPTTHE> <C-AREA> <CASREP> FOR <OPTTHE> <SHIPNAME> BE RESOLVED

2) <WHEN+LL> <OPTTHE> <C-AREA> <CASREP> FROM <OPTTHE> <SHIPNAME> BE RESOLVED.

In this case both the templates are used as input to the acoustic verification system. The final answer is the one that gives the highest acoustic score. For computing the acoustic score, the selected templates are represented as a FSN in terms of the same word HMMs that were used in the speech recognizer. This FSN is used for constraining the search space of a speech recognizer that runs on the original acoustic representation of the uttered sentence.

4. Experimental Results

The semantic postprocessor was tested using the speech recognizer arranged in different accuracy conditions. Results are summarized in Figures 1 and 2. Different word accuracies were simulated by using various phonetic unit models and the two covering grammars (i.e. NG and WP). The experiments were performed on a set of 300 test sentences known as the February 89 test set (Pallett, 1989) The word accuracy, defined as

$$\left[1-\frac{\text{word insertions}+\text{word deletions}+\text{word substitutions}}{\text{number of words uttered}}\right]\times 100 \quad (3)$$

was computed using a standard program that provides an alignment of the recognized sentence with a reference string of words. Fig. 1 shows the word accuracy after the semantic postprocessing versus the original word accuracy of the recognizer using the word pair grammar. With the worst recognizer, that gives a word accuracy of 61.3%, the effect of the semantic postprocessing is to increase the word accuracy to 70.4%. The best recognizer gives a word accuracy of 94.9% and, after the postprocessing, the corrected strings show a word accuracy of 97.7%, corresponding to a 55% reduction in the word error rate. Fig. 2 reports the semantic accuracy versus the original sentence accuracy of the various recognizers. Sentence accuracy is computed as the percent of correct sentences, namely the percent of sentences for which the recognized sequence of words corresponds the uttered sequence. Semantic accuracy is the percent of sentences for which both the sentence generation template and the values of the semantic variables are correctly decoded, after the semantic postprocessing. With the best recognizer the sentence accuracy is 70.7% while the semantic accuracy is 94.7%.

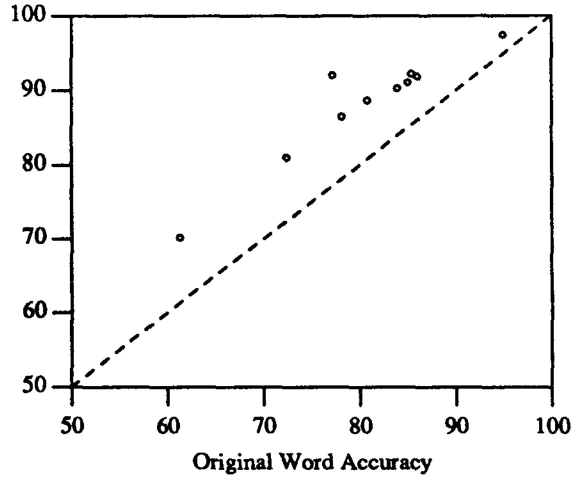

Figure 1. Word accuracy after semantic postprocessing

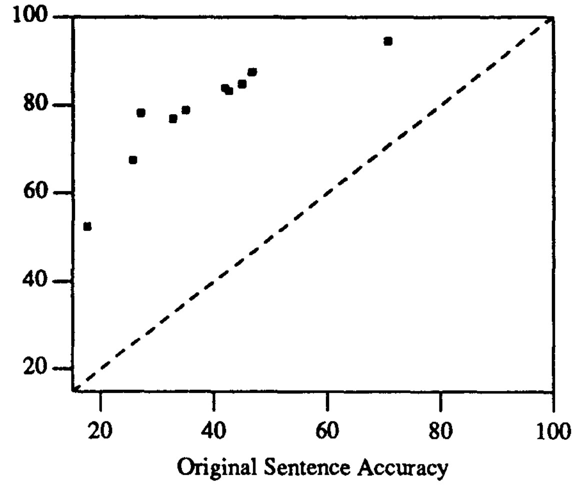

Figure 2. Semantic accuracy after semantic postprocessing

When using acoustic verification instead of simple phonetic verification, as described in section 3.2, better word and sentence accuracy can be obtained with the same test data. Using a NG covering grammar, the final word accuracy is 97.7% and the sentence accuracy is 91.0% (instead of 92.3% and 67.0%, obtained using phonetic verification). With a WP covering grammar the word accuracy is 98.6% and the sentence accuracy is 92% (instead of 97.7% and 86.3% with phonetic verification). The small difference in the accuracy between the NG and the WP case shows the rebustness introduced into the system by the semantic postprocessing, especially when acoustic verification is peformed.

5. Summary

For most speech recognition and understanding tasks, the syntactic and semantic knowledge for the task is often represented in an integrated manner with a finite state network. However for more ambitious tasks, the FSN representation can become so large that performing speech recognition using such an FSN becomes computationally prohibitive. One way to circumvent this difficulty is to factor the language constraints such that speech decoding is accomplished using a covering grammar with a smaller FSN representation and language decoding is accomplished by imposing the complete set of task constraints in a post-processing mode using multiple word and string hypotheses generated from the speech decoder as input. When testing on the DARPA resource management task using the word-pair grammar, we found (Lee, 1990/2) that most of the word errors involve short function words (60% of the errors, e.g. *a, the, in*) and confusions among morphological variants of the same lexeme (20% of the errors, e.g. *six vs. sixth*). These errors are not easily resolved on the acoustic level, however they can easily be corrected with a simple set of syntactic and semantic rules operating in a post-processing mode.

The language constraint factoring scheme has been shown efficient and effective. For the DARPA RMT, we found that the proposed semantic post-processor improves both the word accuracy and the semantic accuracy significantly. However in the current implementation, no acoustic information is used in disambiguating words; only the pronunciations of words are used to verify the values of the semantic variables in cases when there is semantic ambiguity in finding the best matching string. The performance can further be improved if the acoustic matching information used in the recognition process is incorporated into the language decoding process.

6. Acknowledgements

The authors gratefully acknowledge the helpful advice and consultation provided by K.-Y. Su and K. Church. The authors are also thankful to J.L. Gauvain for the implementation of the acoustic verification module.

REFERENCES

1. S. Austin, C. Barry, Y.-L., Chow, A. Derr, O. Kimball, F. Kubala, J. Makhoul, P. Placeway, W. Russell, R. Schwartz, G. Yu, "Improved HMM Models forr High Performance Speech Recognition," *Proc. DARPA Speech and Natural Language Workshop,* Somerset, PA, June 1990.
2. J. K. Baker, "The DRAGON System - An Overview," *IEEE Trans. Acoust. Speech, and Signal Process.,* vol. ASSP-23, pp 24-29, Feb. 1975.
3. M. K. Brown, J. G. Wilpon, "Automatic Generation of Lexical and Grammatical Constraints for Speech Recognition," *Proc. 1990 IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing, Albuquerque, New Mexico, pp. 733-736, April 1990.*
4. G. Hendrix, E. Sacerdoti, D. Sagalowicz, J. Slocum, "Developing a Natural Lanaguge Interface to Complex Data," *ACM Translations on Database Systems* 3:2 pp. 105-147, 1978.
5. X. Huang, F. Alleva, S. Hayamizu, H. W. Hon, M. Y. Hwang, K. F. Lee, "Improved Hidden Markov Modeling for Speaker-Independent Continuous Speech Recognition," *Proc. DARPA Speech and Natural Language Workshop,* Somerset, PA, June 1990.
6. C.-H. Lee, L. R. Rabiner, R. Pieraccini and J. G. Wilpon, "Acoustic Modeling for Large Speech Recognition," *Computer, Speech and Language,* 4, pp. 127-165, 1990.

7. C.-H. Lee, E. P. Giachin, L. R. Rabiner, R. Pieraccini and A. E. Rosenberg, "Improved Acoustic Modeling for Continuous Speech Recognition," *Proc. DARPA Speech and Natural Language Workshop*, Somerset, PA, June 1990.

8. V. I. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions, and Reversals," *Sov. Phys.-Dokl.*, vol. 10, pp. 707-710, 1966.

9. S. E. Levinson, K. L. Shipley, "A Conversational Mode Airline Reservation System Using Speech Input and Output," *BSTJ 59* pp. 119-137, 1980.

10. S. E. Levinson, A. Ljolje, L. G. Miller, "Large Vocabulary Speech Recognition Using a Hidden Markov Model for Acoustic/Phonetic Classification," *Proc. 1988 IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, New York, NY, pp. 505-508, April 1988.

11. S. E. Levinson, M. Y. Liberman, A. Ljolje, L. G. Miller, "Speaker Independent Phonetic Transcription of Fluent Speech for Large Vocabulary Speech Recognition," *Proc. of February 1989 DARPA Speech and Natural Language Workshop* pp. 75-80, Philadelphia, PA, February 21-23, 1989.

12. B. T. Lowerre, D. R. Reddy, "The HARPY Speech Understanding System," Ch. 15 in *Trends in Speech Recognition* W. A. Lea, Ed. Prentice-Hall, pp. 340-360, 1980.

13. H. Murveit, M. Weintraub, M. Cohen, "Training Set Issues in SRI's DECIPHER Speech Recognition System," *Proc. DARPA Speech and Natural Language Workshop*, Somerset, PA, June 1990.

14. D. S. Pallett, "Speech Results on Resource Management Task," *Proc. of February 1989 DARPA Speech and Natural Language Workshop* pp. 18-24, Philadelphia, PA, February 21-23, 1989.

15. R. Pieraccini, C.-H. Lee, E. Giachin, L. R. Rabiner, "Implementation Aspects of Large Vocabulary Recognition Based on Intraword and Interword Phonetic Units," *Proc. Third Joint DARPA Speech and Natural Language Workshop*, Somerset, PA, June 1990.

16. D. B., Paul "The Lincoln Tied-Mixture HMM Continuous Speech Recognizer," *Proc. DARPA Speech and Natural Language Workshop*, Somerset, PA, June 1990.

17. P. J. Price, W. Fisher, J. Bernstein, D. Pallett, "The DARPA 1000-Word Resource Management Database for Continuous Speech Recognition," *Proc. 1988 IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, New York, NY, pp. 651-654, April 1988.

18. L. R. Rabiner, "A Tutorial on Hidden Markov Models, and Selected Applications in Speech Recognition," Proc. IEEE, Vol. 77, No. 2, pp. 257-286, Feb. 1989.

19. D. R. Reddy, et al., "Speech Understanding Systems: Final Report," *Computer Science Department, Carnegie Mellon University, 1977.*

20. W. Woods, et al., "Speech Understanding Systems: Final Technical Progress Report," *Bolt Beranek and Newman, Inc. Report No. 3438*, Cambridge, MA., 1976.