Summer 1990

Encoding Phonetic Knowledge for Use in Hidden Markov Models of Speech Recognition

Danming Qian
Old Dominion University

ENCODING PHONETIC KNOWLEDGE FOR USE IN HIDDEN MARKOV MODELS OF SPEECH RECOGNITION

By

Danming Qian
B.S.E.E. August 1988, Old Dominion University

A Thesis Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

ELECTRICAL AND COMPUTER ENGINEERING

OLD DOMINION UNIVERSITY
June, 1990

Approved by:

Stephen A. Zahorian (Director)

Oscar A. Gonzalez

David Livingston

ENCODING PHONETIC KNOWLEDGE FOR USE IN HIDDEN MARKOV MODELS OF SPEECH RECOGNITION

By

Danming Qian

Stephen A. Zahorian, Advisor

Old Dominion University
Department of Electrical and Computer Engineering
Norfolk, Virginia 23529

June 1990

ABSTRACT

ENCODING PHONETIC KNOWLEDGE FOR
USE IN HIDDEN MARKOV MODELS OF SPEECH RECOGNITION

Danming Qian
Old Dominion University, 1990
Director Dr. Stephen A. Zahorian

Hidden Markov models (HMM's) have achieved considerable success for isolated-word speaker-independent automatic speech recognition. However, the performance of an HMM algorithm is limited by its inability to discriminate between similar sounding words. The problem arises because all differences between speech patterns are treated as equally important. Thus the algorithm is particularly susceptible to confusions caused by phonetically-irrelevant differences. This thesis presents two types of preprocessing schemes as candidates for improving HMM performance. The aim is to maximize the differences between phonologically-distinct speech sounds while minimizing the effect of variations in phonologically-equivalent speech sounds. The preprocessors presented are a discrete cosine transformation (DCT) and linear discriminant analysis type transformation (LDA).

The HMM used in this investigation is a five-state, left-to-right structure. All the experiments were performed with either 30 or 99 highly confusable words from a CVC isolated-word data base. Computations were performed on UNIX

SUN work stations. All words were hand labeled in terms of acoustic-phonetic segments. The DCT preprocessing, a block transform encoding with data-independent basis vectors, was not found to be successful for improving overall word recognition performance. In contrast, the LDA preprocessing method did improve HMM word recognition accuracy. The LDA basis vectors were computed from signal statistics so as to maximize the ratio of between to within phonetic class data variance. The LDA technique requires phonetically segmented data for training. Using speaker-independent word recognition tests, i.e., one set of speakers for training and another set of speakers for testing, the LDA method reduced HMM word errors over 45%. Results show that discrimination between similar sounding words can be greatly improved.

The results of the research conducted in this study not only gives additional insights into the basic operation of hidden Markov modeling for speech recognition, but also could potentially be applied to large vocabulary continuous-speech speaker-independent speech recognition. It shows that significant improvements in speech recognition system performance may be achieved by better acoustic-phonetic modeling.

ACKNOWLEDGMENT

I would like to thank my advisor, Dr. Stephen A. Zahorian, for his invaluable motivation and guidance. His continuous encouragement and availability were greatly appreciated. I also would like to thank the other members of my committee, Dr. David Livingston, and Dr. Oscar Gonzalez for their generous assistance and time.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION AND PROBLEM DEFINITION

Hidden Markov models (HMM's) based speech recognition plays a predominant role in modern speech recognition system design. The most important factors to its success include a capability of dealing with the random nature of speech, a mechanism for incorporating high level speech knowledge (syntax and grammar), and an efficient training and evaluation algorithm. However, the low level acoustic phonetic performance is generally poor.

The fundamental problem in automatic speech recognition (ASR) is the inherent variability of speech signals. No word is ever produced in exactly the same way on separate occasions. For a single speaker, a major variation is in the lengths of words. Hidden Markov model based techniques are able to compensate for time variations and are thus able to produce useful recognition accuracies on vocabularies that contain easily distinguishable words. However, for vocabularies that contain similar sounding words, recognition performance is generally poor.

Of course it is not surprising that recognition algorithms experience some difficulty distinguishing similar sounding words, which even tax the limits of human listeners.

However, there is considerable room for improvement in automatic speech recognition (ASR). A major limitation of the typical HMM training algorithm for words is that the network structure for each word model is not learned but rather specified a priori before training. The word models are created using phonetic dictionaries and complex phonological rules to specify each word as a network of subword units such as phonemes. Because of coarticulation effects, the acoustic properties of phonemes are quite different depending on the preceding and following phoneme. Thus phoneme and word modeling can be poor if a single context-independent model is used for each phoneme. If context-dependent phoneme models are created, excessive computations are required during recognition.

Our goal is to form a speech feature space such that phonemes are well-clustered independently of context. For example, the final consonant segment of "BAT" and "PAT" should be well-modeled by a single "T" model. The "T" model will also be easily distinguished from other phoneme models. With irrelevant differences between phonetically-equivalent word components eliminated the network can focus attention on real acoustic-phonetic differences.

The solution to this problem is a speech pattern matching algorithm focused on those regions of a pattern that serve to distinguish it from other similar sounding patterns. Since phonemes are the basic units which distinguish words in a language, it is expected that a transform which enhances

phonetic discriminability will greatly improve the overall recognition rate. In general, phonologically-equivalent speech segments map to the same region in the phonetic feature space.

To test any method for improving HMM speech recognition, we must verify that the basic HMM is performing properly. Therefore, in addition to describing preprocessing methods, we present results of parametric studies which give performance while varying the number of states in HMM's and number of training tokens per word.

1.1 Comparison between HMM and Conventional Pattern Recognition Systems

There currently exist two standard approaches to isolated-word recognition, dynamic time warping (DTW) recognizers and hidden Markov model (HMM) recognizers. Figure 1-1 shows these two recognizers.

The signal preprocessing part is the same for both DTW and HMM basis recognizers. The input speech signal is bandpass-filtered and sampled. Speech features are extracted from each frame after the signal preprocessing. For the DTW recognizer, the features are time aligned and output to the next stage. For a HMM recognizer, the feature vectors are quantized and the code indices are output to the next stage. The DTW algorithm (distance measure) or HMM algorithm (probability measure) is used to form the reference patterns. The decision rule for the DTW is based on a distance score,

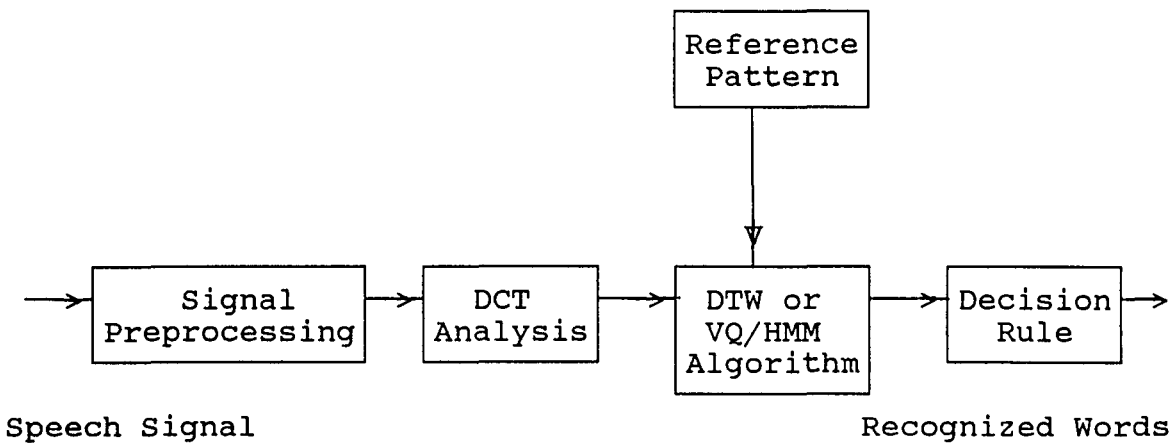


Figure 1-1. Block diagram of DTW and HMM recognizers.

while the decision rule for HMM is based on a probability score. A DTW recognizer usually needs more than ten templates per word, while a HMM recognizer only needs one model per word to achieve satisfactory results

1.2 Hidden Markov Models: A Representation of Speech

It is known that information in the speech signal is encoded in the temporal variation of its short-duration power spectrum. To decode the signal requires techniques for both estimation of power spectra and tracking their changes in time. A hidden Markov model (HMM) is a powerful technique capable of robust modeling of speech. A HMM is particularly suitable for describing speech events. It is capable of dealing with the random nature of speech signals and with talking rate variations. High level speech knowledge can also be incorporated in the model. HMM's consist of two stochastic processes that enable the modeling not only of acoustic phenomena but also of timescale distortions. Furthermore, efficient algorithms exist for accurate estimation of HMM parameters. Unlike non-parametric approaches, the forward-backward re-estimation algorithm for HMM's is an instance of the estimate maximize (EM) algorithm. As such, every iteration of the algorithm results in an imposed set of model parameters. HMM's are a succinct representation of speech events, therefore, they require less storage than many other strategies (Deng and Lenning, 1988).

1.3 Summary and Thesis Outline

In the previous section, we have claimed that a HMM has many advantages for ASR over conventional pattern recognition methods. However, the main limitation of a HMM is that it requires large amounts of training data since it is a statistical model. It is desired that the amount of the training data required be as small as possible, while maintaining high recognition accuracy for test data.

It has been shown that a phonetically sensitive transformation of speech features can yield significant improvement in speech recognition performance (Doddington, 1989). This (linear) transformation of the speech feature vector is designed to discriminate against out-of-class confusion data and is a function of phonetic state. A similar approach is investigated in this thesis. We believe the "phonetic enhancement" preprocessing for HMM-based recognition will overcome some of the inherent disadvantages while maintaining advantages to improve word recognition capability.

The remainder of the thesis is organized as follows. Chapter two gives background information on hidden Markov modeling of speech along with some experimental results used to verify the operation of the HMM. Basic aspects of cluster theory, vector quantization, and codebook implementation, essential to discrete HMM's, are also given in chapter two. In chapter three, we discuss two different kinds of feature preprocessing techniques--the discrete cosine transformation (DCT) and linear discriminant analysis type transformation

(LDA). The conclusion is included in the last chapter along with suggestions for further research.

CHAPTER 2

BACKGROUND AND OVERVIEW

Hidden Markov models (HMM's) were first described in the classic paper by Baum (1972). Shortly afterwards, they were used for automatic speech recognition by many researchers. It has only been in the past few years, however, that HMM's have become the predominant approach to speech recognition, superseding dynamic time warping.

In this chapter, we will first define HMM's and show how the speech recognition problem can be formulated as a HMM problem. Then we will present an algorithm for training, evaluating, and decoding HMM's as well as an implementation issue for HMM's. Finally, we will present results of isolated-word recognition based on either 99 or 30 CVC word data base and analyse the results. The results will be used to verify correct operation of our HMM implementation. We follow the algorithm presented by Rabiner (1983).

2.1 Definition of a Hidden Markov Model

A probabilistic function of a (hidden) Markov chain is a stochastic process generated by two interrelated mechanisms. An underlying Markov chain having a finite number of states, and a set of random functions, one of which is associated with each state. At discrete instants of time, the process is

assumed to be in some state and an observation is generated by the random function corresponding to the current state. The underlying Markov chain then changes states according to its transition probability matrix. The observer sees only the output of the random function associated with each state and cannot directly observe the states of the underlying Markov chain; hence the term hidden Markov model.

It is quite natural to think of the speech signal as being generated by such a process. We can imagine the states as different positions of the vocal organs, and the outputs as the acoustical observation associated with each articulatory position (Thomes, 1983). In each state, a short (in time) signal is produced that has one of a finite number of prototypical spectra depending on the state. Thus the power spectra of a short interval of the speech signal is determined solely by the current state of the model, while the variation of the spectral composition of the signal with time is governed predominantly by the probabilistic state transition law of the underlying Markov chain. By allowing random transitions and outputs, we enable the model to cope with subtle variations in pronunciation and timing.

The most natural unit of speech is the word. We can use a model similar to that depicted in figure 2-1 to represent a word '<u>bird</u>'. From this figure, we can see that a state could correspond to some phonetic event, and events could be skipped (figure 2-1). Since the amount of training and storage is enormous for large-vocabulary recognition,

researchers have used phone or phoneme models. An example of a phoneme model is shown in figure 2-2. The three states with self-loops could represent the transition into the phoneme, the steady state portion, and the transition out of the phoneme (figure 2-2).

There are two assumptions in a first-order Hidden Markov Model.

1. Markov assumption: The probability that the Markov chain is in a particular state at time t+1 depends only on the state of the Markov chain at time t and is conditionally independent of the past.

2. Output independence assumption: The probability that a particular symbol will be emitted at time t depends only on the state at that time.

Although these assumptions severely limit the memory of first order HMM's, they reduce the number of parameters. As we will see, they also make learning and decoding algorithms more efficient.

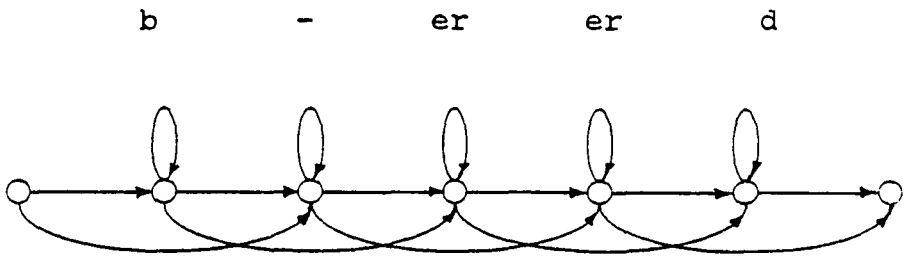


Figure 2-1. A HMM representation of a word.

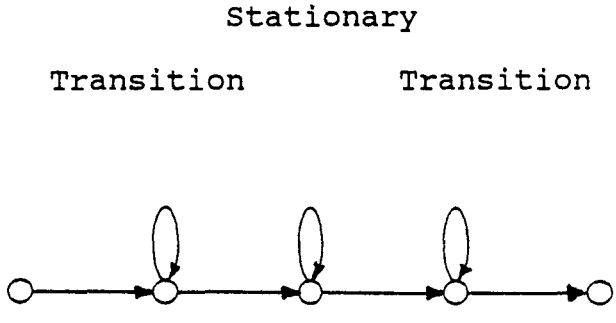


Figure 2-2. A HMM representation of a phoneme.

2.2 Three HMM Problems and Their Solutions

There are three major problems involved in implement the HMM's. These three problems are: (1) find the probability that the model generated the observations by giving a model and a sequence of observations (testing problem); (2) find the most likely state sequences in the model that produced the observation sequence by giving a model and a sequence of observations (decoding problem); and, (3) find the model's parameter be so that it has a high probability of generating the observation sequences by giving a model and a set of observations (training problem).

If we could solve the evaluation problem, we would have a way of scoring the match between a model and an observation sequence, which could be used for isolated-word recognition. If we could solve the decoding problem, we could find the best matching state sequence given an observation sequence, which could be used for continuous speech recognition. Most importantly, if we could solve the training problem, we would optimally derive model parameters for each word model. In this section, we will outline the solutions for these three problems.

2.2.1 The Evaluation Problem

The most straightforward way of calculating the probability of the observation sequence O, given a model M or $P(O|M)$ is through enumerating every possible state sequence of length T (the number of observations)

$$P(O|q,M)=b_{i1}(O_1)b_{i2}(O_2)\ .....b_{iT}(O_T)\ , \qquad (2\text{-}1)$$

where $q=i_1i_2\ .....i_T$ is a state sequence.

The probability of such a state sequence q is

$$P(q|M)=\pi_{i1}a_{i1i2}a_{i2i3}.....a_{iT-1iT}\ . \qquad (2\text{-}2)$$

The probability that O and q occur simultaneously is

$$P(O,q|M)=P(O|q,M)*P(q|M)\ . \qquad (2\text{-}3)$$

The probability of O is obtained by summing point probabilities over all possible state sequences

$$P(O|M)=\sum_{\text{all } T} P(O|q,M)*P(q|M)=\sum_{i1i2\ldots i_T} \pi_{i1}b_{i1}(O_1)a_{i1i2}b_{i2}(O_2) \ldots\ldots a_{iT-1iT}b_{iT}(O_T)\ . \qquad (2\text{-}4)$$

We can see that the evaluation of equation (2-4) requires enumeration of all paths with length T, which is clearly exponential. Fortunately however, there is an efficient method for computing P. Let us define the function $\alpha_t(i)$ for $1\le t\le T$ as $P(O,q_i$ at $t|M)$ according to the definition $\alpha_1(i)=\pi_i b_i(O_1)$, where $b_i(O_1)$ is understood to mean b_{ik} iff $O_1=q_k$. We then have the following recursive relationship for the "forward probabilities"

$$\alpha_{t+1}(j)=[\sum_{i=1}^{N}\alpha_t(i)a_{ij}]b_j(O_{t+1}). \qquad 1\le t\le T-1 \qquad (2\text{-}5)$$

Similarly, we define another function $\beta_t(j)=P(O|q_t$ at $t,M)$

set $\beta_T(j)=1$ for any j, and use the backward recursion

$$\beta_t(i)=\sum_{j=1}^{N} a_{ij}b_j(O_{t+1})\beta_{t+1}(j), \qquad T-1\geq t\geq 1 \tag{2-6}$$

to compute the "backward probabilities".

$$\text{Thus} \quad P(O|M)=\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j). \tag{2-7}$$

Since $\alpha_T(i)=P(O,q_t|M)$,

$$P(O|M)=\sum_{i=1}^{N} \alpha_T(i), \tag{2-8}$$

therefore $P(O|M)$ can be computed from the forward probabilities alone.

From the above derivations, we can see that the forward-algorithm enables us to evaluate the probability that an observation sequence was generated by a HMM. However, in speech recognition, we need to find $P(M|O)$. By Bayes rule, we have

$$P(M|O)=\frac{P(O|M)*P(M)}{P(O)} . \tag{2-9}$$

2.2.2 The Decoding Problem

Finding the optimal state sequence could be used for segmentation and recognition of speech. When we compute

$P(O|M)$ with forward-backward algorithm (section 2.2.1), we are including the probabilities of all possible state sequences that may have generated O. Unfortunately, by definition, the state sequence is hidden in a HMM. The best we can do is to produce the state sequence that has the highest probability of being taken while generating the observation sequence. To do this, we only need to modify the forward pass slightly. In the forward pass, we summed all the probabilities that came together. Now we need to choose the maximum

$$\phi_t(j)=\max[\phi_{t-1}(i)a_{ij}]b_j(O_t), \quad \text{for } 2\le t\le T \quad 1\le j\le N \quad (2\text{-}10)$$

where $\phi_1(i)=\pi_i b_i(O_1)$, for $1\le i\le N$

and

$$\Phi_t(j)=i^* \ ,$$

where i^* is a choice of an index i that maximize $\phi_{t-1}(i)$.

This algorithm is known as the Viterbi algorithm, it uncovers the most likely state sequence. It can be used for segmentation annotation and recognition.

2.2.3 The Training Problem

This iterative procedure can be used to maximize the probability of the observation sequence given the model. We can use the forward and backward probabilities to formulate a solution to the problem of training by parameter estimation.

The expected number of transitions μ_{ij} from q_i to q_j conditioned on the observation sequence is just

$$\mu_{ij} = \frac{1}{P} \sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \quad . \tag{2-11}$$

The expected number of transitions μ_i out of q_i given O is

$$\mu_i = \sum_{j=1}^{N} \mu_{ij} = \frac{1}{P} \sum_{t=1}^{T-1} \alpha_t(i) \beta_t(i) \quad . \tag{2-12}$$

The ratio μ_{ij}/μ_i is an estimate of the probability of the state q_j, given that the previous state was q_i. It may be taken as a new estimate of a_{ij}

$$\bar{a}_{ij} = \frac{\mu_{ij}}{\mu_i} = \frac{\sum_{i=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{t=1}^{T-1} \alpha_t(i) \beta_t(t)} \quad . \tag{2-13}$$

Similarly, the new estimate of b_{jk}, the frequency of occurrence of v_k in q_j relative to the frequency of occurrence of any symbol in state q_j, is

$$\bar{b}_{jk} = \frac{\sum_{O_t=v_k} \alpha_t(j) \beta_t(j)}{\sum_{t=1}^{T} \alpha_t(j) \beta_t(j)} \quad . \tag{2-14}$$

It has been proved that these re-estimations are guaranteed to increase P, except at a critical point (Rabiner and Juang, 1986). This is one of the main reasons that a HMM is superior to other classification methods.

2.3 Implementation Issues

Practical problems.

1. Any of the methods presented here for either the classification or the training problem requires evaluation of $\alpha_t(i)$ and $\beta_t(i)$ for $1 \le t \le T$ and $1 \le i \le N$. It is clear that as $T \to \infty$, $\alpha_t(i) \to 0$ and $\beta_T(i) \to 0$ in exponential form. That will result in an underflow on any floating point computers.

2. Finite training sets can cause some of the probabilities of the observation to be zero. This phenomenon is fatal to a classification task.

2.3.1 Initialization

Although the Baum-Welch algorithm is guaranteed to reach a critical point of $P(O|M)$, alternative starting values of A (transition probability matrix) and B (output probability matrix) matrices could yield a model with higher (or lower) values of $P(O|M)$. Additionally, the initialization must satisfy the constraints:

$$\sum_{j=1}^{N} a_{ij} = 1 \quad \text{for} \quad i=1,2,\ldots,N, \tag{2-15}$$

$$\sum_{k=1}^{M} b_j(k) = 1 \quad \text{for} \quad j=1,2,\ldots,N, \tag{2-16}$$

where N is the number of states,

and M is number of codewords.

We used five-state, no transitions that skip states, and left-to-right models in all our experiments. Pilot experiments were used to arrive at these choices (Rabiner et. al., 1983).

We initialized the transition matrix A as

$$A = \begin{bmatrix} .5 & .5 & 0 & 0 & 0 \\ 0 & .5 & .5 & 0 & 0 \\ 0 & 0 & .5 & .5 & 0 \\ 0 & 0 & 0 & .5 & .5 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

For the output B matrix, we assigned the initial values according to the assumed frequency of the codewords appearing in that state, as follows.

1. Equally divide each stimulus into 5 blocks which, are approximately associated with the 5 states for the model. The last block may be longer than the previous ones.
2. For each block, find the frequency of codewords which appear there.
3. For those codewords which never appear in the training data, fill the positions with $\epsilon=10^{-5}$. (The reason is explained in section 2.3.3)

In this way, we can optimize the initial conditions for the B matrix. It may not increase the recognition rate, but it converges much faster when we train the HMM's.

2.3.2 Scaling

The principle is to multiply $\alpha_t(i)$ by some scaling coefficients independent of i, so that it remains within the dynamic range of the computer for $1 \le t \le T$. The same procedure is used for $\beta_t(i)$. At the end of the computations, we remove the total effect of the scaling factors. We multiply $\alpha_t(i)$ by a scaling coefficient C_t ,

say $C_t = \left[\sum_{i=1}^{N} \alpha_t(i) \right]^{-1}$, so that $\sum_{i=1}^{N} C_t \alpha_t(i) = 1$, for $1 \le t \le T$.

We also multiply $\beta_t(i)$ by the same factor $C_t \beta_t(i)$ for $T \ge t \ge 1$ and $1 \le i \le N$.

Equation (2-13) becomes

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} C_t \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) D_{t+1}}{\sum_{t=1}^{T-1} \sum_{l=1}^{N} C_t \alpha_t(i) a_{il} b_l(O_{t+1}) \beta_{t+1}(l) D_{t+1}} \quad . \quad (2\text{-}17)$$

Since $C_t D_{t+1} = \prod_{\tau=1}^{T} C_\tau$ can be factored out and canceled, equation (2-17) has the correct value of $\bar{a}_{ij}$.

P can be recovered from the scale factors as follows:

$$\log P = - \sum_{t=1}^{T} \log C_t \quad . \quad (2\text{-}18)$$

Later, we will see that the combination of maximizing log P and this scaling technique simplifies the solution of the left-to-right Markov modeling problems as well.

2.3.3 Finite Training Set

In reality, the observation sequence will always be finite. Suppose a given training sequence of length T results in $b_{jk}=0$. Further assume that we are subsequently asked to compute the probability that a new observation sequence was generated by our model. It is possible that $\alpha_{t-1}a_{ij}$ is nonzero for only one value of j and that $O_t=v_k$, thus forcing $\alpha_t(j)=0$. Therefore, the probability of the observation then becomes zero. This can be solved by a set of new constraints $a_{ij} \geq \epsilon > 0$, $b_{jk} \geq \epsilon > 0$. The modified Baum-Welch algorithm is as follows. Evaluate the B matrix using the re-estimation formulas. Assume that some set of the parameters in the jth row of B violates the constraint, so that $b_{jki} < \epsilon$ for $1 \leq i \leq 1$. Then set $b_{jki} = \epsilon$ and readjust the remaining parameters according to the following equation

$$\bar{b}_{jk} = (1-1\epsilon) \frac{b_{jk}}{\sum_{i=1}^{N-1} b_{ji}} . \qquad (2\text{-}19)$$

After performing the operation of equation (2-19), the resulting B matrix is the optimal updated matrix with respect to the desired constraints.

2.3.4 Left-to-right HMM's

In our isolated-word recognition application, we are interested in non-ergodic models where we impose constraints on the state transition matrix. For example, figure 2-3 and figure 2-4 show two examples of non-ergodic HMM's. For these cases, the state transition matrix is upper triangular. Such models have been called left-to-right models since the state sequence that produced the observation sequence must always proceed from the left-most states to the right-most states. Such left-to-right models inherently impose a temporal order to the HMM, since lower numbered states account for observations occurring prior to those for higher numbered states.

For isolated-word recognition, it has been shown that a simplified model, the left-to-right model, can be used to achieve the same or better results than with unconstrained models (Rabiner, Levinson, and Sondhi, 1983). The left-to-right model has the following properties.

1. The first observation is produced while the Markov chain is in the starting state called q_1.
2. The last observation is generated while the Markov chain is in the final state.
3. Once the Markov chain leaves a state, that state cannot be revisited.

Condition 1 can be satisfied by setting $\pi=(1,0,\ldots,0)$ and not reestimating it. Condition 2 can be satisfied by setting $\beta_T(j)=1$ for $j=N$ and 0 otherwise. Condition 3 can be satisfied

by setting $a_{ij}=0$ for $j<i$. Since the Baum-Welch algorithm computes the frequency of occurrence of various events, all we need to do is to compute these frequencies of occurrence in each sequence separately and add them together. Thus the new re-estimation formulas become

$$\bar{a}_{ij} = \frac{\sum_{k=1}^{K} \sum_{t=1}^{T-1} \alpha_t^k(i)\, a_{ij}\, b_j(O_{t+1}^k)\, \beta_{t+1}^k(j)}{\sum_{k=1}^{K} \sum_{t=1}^{T-1} \frac{1}{C_t}\, \alpha_t^k(i)\, \beta_t^k(i)} , \quad (2\text{-}20)$$

$$\bar{b}_{ij} = \frac{\sum_{k=1}^{K} \sum_{O(k)=v_k} \frac{1}{C_t}\, \alpha_t^k(i)\, \beta_t^k(i)}{\sum_{k=1}^{K} \sum_{t=1}^{T} \frac{1}{C_t}\, \alpha_t^k(i)\, \beta_t^k(i)} , \quad (2\text{-}21)$$

and

$$P = \prod_{k=1}^{K} \text{Prob}(O^k|M) = \prod_{k=1}^{K} P_k . \quad (2\text{-}22)$$

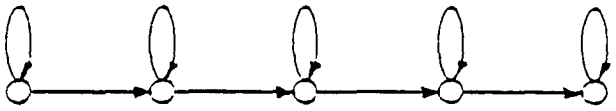


Figure 2-3. A left-to-right hidden Markov model with starting state one, and ending state five.

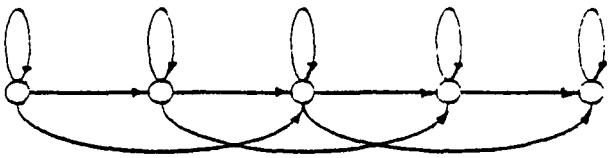


Figure 2-4. A left-to-right hidden Markov model with transitions that skip states.

2.3.5 Brief Summary

The experimentation with various forms of the Markov models used in the recognizer showed conclusively that (Levinson et. al., 1983):

1. Constrained models (with constrained transition matrices) perform consistently better than unconstrained models.
2. A finite minimum constraint on the state symbol probability matrix is a necessity for good system performance.
3. The effects of different random starting values for the HMM parameters are negligible in evaluating overall recognizer performance.
4. The required number of states in each word HMM needs to be on the order of five. More states do not lead to significant improvements in performance.
5. Parallel HMM structures yield no real improvements over cascade structures, thereby indicating that an equivalent of multiple HMM's is not readily obtainable by simply changing the model structure.
6. The Viterbi scoring and the Baum-Welch scoring of test sequences give essentially identical performance.

2.4 Vector Quantization

Vector quantizers are used in image and speech transmission systems to reduce the amount of data without losing important information. They group similar data into a cluster. The number of clusters can be pre-specified or may be allowed to grow according to some criteria.

The vector quantization idea is depicted in figure 3-1. A set of training speech sequences is first used to generate the codebook. The speech is segmented into successive short frames and each frame of speech is represented by a vector of finite dimensionality. Codebook generation requires an iterative process much like a clustering algorithm which involves a large number of spectral comparisons. The generation produces a finite collection of spectral model vectors (codebook) so that the average spectral distortion from all the input vectors to their best match in the codebook is minimized. Each input speech vector is mapped to the codebook entry (codeword) index corresponding to the best matching vector. The vector quantizers we used are based on Euclidean distance measurements.

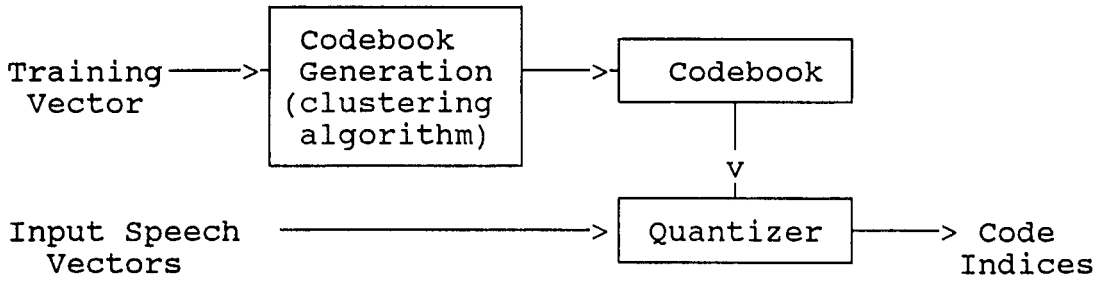


Figure 2-5. Block diagram of vector quantization technique.

2.4.1 Clustering Theory

Sometimes features may separate well enough that the classes occupy essentially nonoverlapping regions in the feature space. Here, it is possible to allow the recognizer to group the training data into categories without supervision. This gives rise to a process known as clustering. Clustering is closely related to the techniques used to create the codebooks used in vector quantization. Two clustering algorithms have been investigated in our research.

1. The K-Means Algorithm

In this algorithm, the initial clusters are characterized only by their means. Every data point is assigned to the cluster whose mean is nearest to it. After all data points have been assigned, the cluster means are recomputed. This process is repeated until no changes in assignments occur.

2. The Farthest Neighbor Algorithm

This method starts with a single cluster consisting of one data point, then the remaining points are tested and the point that is farthest from the initial cluster is chosen as the nucleus of the second cluster. The point that is most remote from these two clusters is made the nucleus of the third cluster. This process continues either until the number of clusters is equal to the number of codewords or until the maximum distance is less than some threshold value. After initialization, the algorithm is iterated as for K-means.

2.4.2 Implementation of Codebook

The idea behind the codebook is to collect samples into groups and to encode the groups. If we consider a sequence of samples as a vector, then the codebook entries are vectors rather than individual samples. Furthermore, the larger the codebook size, the better the representation. This is the basis of vector quantization. Vector quantization has been applied to waveforms as well as to predictor parameters.

To initialize the codebook, we have to make sure the sample vectors are far enough apart so that their correlations are negligible. In order to eliminate the possibility of local maxima, we could use the farthest-neighbor algorithm discussed in previous section or use a splitting method (not discussed here). Since both methods require a long time to initialize the codebook, we used a much simpler method. In particular, points uniformly distributed over the entire input space were chosen as the initial cluster nuclei. The K-means algorithm discussed in section 2.4.1 was iterated until the clusters were well formed, as measured by less than .01% percent change in distortion due to vector quantization. Larger codebooks can have better results, but the time needed to form the codebook increases as well. Typically, we use a codebook size between 32 to 256 codewords. In fact, if too many codewords are used, the test recognition rate will drop due to the insufficient training data, which will be shown in section 2.5.2.2.

2.5 HMM's for Speech Recognition

In the previous sections, we present an approach to speaker-independent isolated-word recognition in which the well known techniques of vector quantization and hidden Markov modeling are combined with a DCT analysis front end. Although the main goal of this study was to evaluate preprocessing for discrete HMM isolated-word recognition, at the start of this project the software required to implement HMM's was not available in the speech lab at Old Dominion University. Therefore the first major undertaking of this study was to implement the needed software using the equations given in this chapter. In the remainder of this chapter we give experimental results for tests to verify the operation of the HMM software. Additionally the experimental results given in this chapter are needed for a control for the results presented in chapter three. In the next section, we briefly discuss the data base we used and present results obtained to test the models. The block diagram of the HMM/VQ isolated-word recognizer used for testing is shown in figure 2-6.

2.5.1 Signal Preprocessing

2.5.1.1 Data Base

Ninety-nine CVC (Consonant-Vowel-Consonant) isolated tokens were recorded for each of 30 native-English talkers. Ten of these 30 talkers were adult males, ten were adult females, and ten were children between the ages of seven and eleven. About 2/3 of these 99 CVC tokens were meaningful

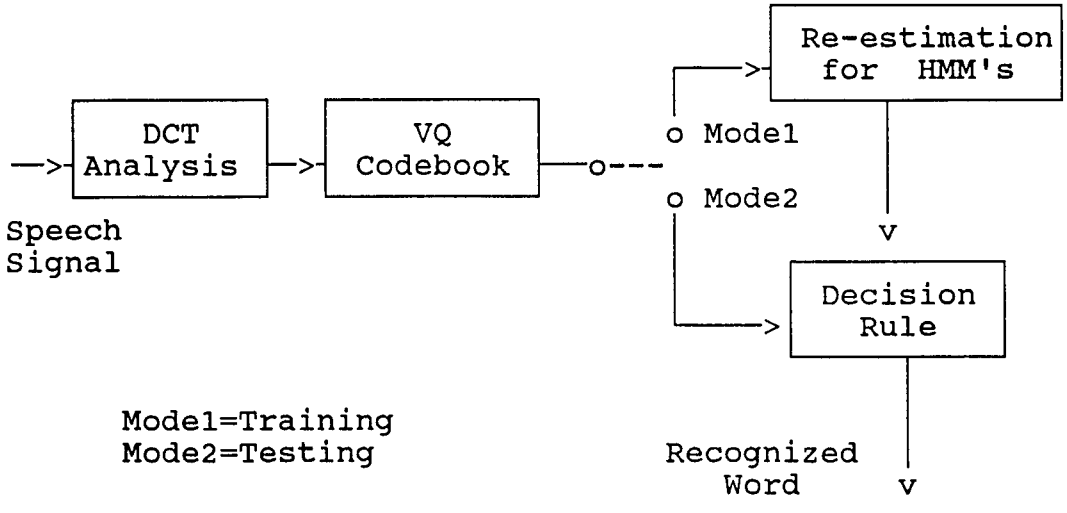


Figure 2-6. Overall block diagram of the HMM/VQ isolated-word recognizer.

words and about 1/3 were nonsense words. Eighty-four of these syllables began with one of the six stop consonants /b,p,d,t,g,k/, and the other 15 syllables began with one of the four consonants /h,l,m,w/. The vowel in each syllable was one of the eleven vowels /aa, iy, uw, er, ih, ae, eh, ao, ah, uh, ow/. Ninety six of these 99 CVC tokens ended with one of the six stops /b,p,d,t,g,k/ and the other three tokens ended with one of the three consonants /v,s,h/. Each initial or final stop was paired with at least one instance of each of the eleven vowels. i.e., each initial or final stop was spoken in eleven vowel contexts. In the appendix, we provide a list of these 99 CVC isolated-words.

In our experiment, due to the limited amount of memory space and the time it takes for training the models, we chose 30 words for most of the experiments. For a few cases, 50 or 99 words were also used.

2.5.1.2 Recording Conditions and Signal Preprocessing

All recording sessions were held in a soundproof room. The typical sound level of speech sounds was approximately 36 dB above the background noise level in the room. Each speech waveform was low-pass filtered at 7.5 kHz with a 6^{th} order Butterworth analog filter prior to sampling at 16 kHz with a 12 bit A/D converter. All the speech files were digitally highpass filtered at 240 Hz with a 62^{nd} order linear phase FIR filter to remove low-level low-frequency noise in the signal.

2.5.1.3 Segmentation

The acoustic regions of all speech files were hand labeled in terms of acoustic segments with the help of an interactive computer waveform editor. The segments labeled included Initial Burst (IB), Initial Transition (IT), Steady Vowel (SV), Final Transition (FT), and Final Burst (FB). The segment labels were not used (except for the starting and ending points of each word) for HMM verification experiments reported in this chapter. These labels were, however, required for the main experiments reported in the next chapter. More information on the data base, recording conditions, signal preprocessing, and labeling is given in Nossair (1989).

2.5.2 HMM's for Isolated-word Recognition

Isolated-word recognition systems are much simpler than continuous speech recognition systems. Many of the techniques developed for isolated-word ASR have been carried over into word-spotting and continuous-speech recognition. Pauses between words simplifies recognition because they make it relatively easy to identify endpoints, and they minimize coarticulation effects between words. In addition, isolated-words tend to be pronounced somewhat more carefully than words spoken in sentences.

2.5.2.1 Training and Testing

HMM training for isolated-words can be implemented directly using the forward-backward algorithm. We could use the forward pass to score the input word against each of the models. The model with the highest probability is chosen as the recognized word. It is not necessary to clip the speech from the beginning and ending silences because these silent intervals will be absorbed in the states of the word models. We trained one discrete HMM for each of word in our data. We tested many HMM topologies, and found the one with five-state, left-to-right, no transitions that skip states to be the best for our data base (figure 2-3).

The Viterbi algorithm could have been used for recognition. However, for data bases consisting of a relatively small number of isolated-words (such as the one used in this study), an exhaustive search based on the forward algorithm is preferred because the probability from the Viterbi algorithm is only an estimate of the correct probability.

Steps to perform isolated-word recognition:

1. Build a HMM for each word in the vocabulary using training tokens.
2. Calculate $P(O|M)$ according to the procedure discussed in section 2.2.1.
3. Choose the word whose model probability is the highest.

2.5.2.2 Evaluation of Experiments and Results

All the experimental results shown in this section may be divided into two different phases: the training phase and the testing phase. The HMM's are developed in the training phase from a subset of the total available data. Evaluation results can be obtained from this training data with the HMM's. For speech recognition, we are more interested in the test results, obtained from tokens not used to created the HMM's, since such results give a much better measure of the ability of the HMM's to generalize. In the following discussion, we focus mainly on the test results. However, at some points, we also present the training results for comparison purpose. In general, classification based on the test data is worse than classification of the training data. Training data does not prepare us for some of the phenomena observed in the test data. However, as the data base for training become very large, the two results should be very similar.

The discrete distribution speaker-independent isolated-word HMM recognizer was used for all experiments. We used five state, linear, left-to-right word models with no transitions that skip states throughout the experiments. We used either 30, 50 or 99 CVC isolated-words for each experiment. Due to some mispronounced words, a total of 2909 tokens were avaliabled. In all cases data from a subset of speakers was used for training (either 15 or 20 speakers); data from the remaining 15 or 10 speakers were used for

testing. Therefore all test results are for speaker-independent ASR.

Several tests were conducted. Table 2-1 shows the results using 99 words and 15 speakers for training and 15 speakers for testing. Since these 99 words are very similar sounding, a test recognition rate of 44.01% was obtained with 256 codewords. Results are also given with the training and testing speakers interchanged, to verify that the results were not overly biased by the particular selection of training and test speakers. The results in table 2-1 show that both testing and training recognition rates are very close for both speaker arrangements, which means no big bias exists in our data base.

Table 2-2 gives recognition rates obtained with two different kinds of HMM's, one with and one without the transitions to the state after the following states. Those results are almost the same, so we chose the model without the transitions that skip states in all our later experiments to save computational time. Table 2-3 indicates the effects of the number of training speakers (and thus training tokens for each word) using the first 30 words of our 99 word data base. Results are given both for 20 training speakers and 10 test speakers and for 15 training speakers and 15 test speakers. As expected as the number of training speakers decreases, training results improve and test results degrade. However, in both cases the training results are much higher than the test results, thus indicating that the data base for training

was not really large enough. It is probably the case that a far larger data base is required to achieve very high speech recognition performance than is generally acknowledged. With fewer codewords, the models are very poor, and some of the testing data may even match the models better than the training data. This will result in testing rates higher than training rates. The models become more and more complex with more codewords, since many more probabilities must be estimated. This in turn implies that much more training data is required. Since the amount of data for training is fixed, the testing results may deteriorate as the number of codewords increase. The training results improve as the number of model parameters increases, eventually reaching 100%. These results are as expected, because all computations are based on statistical pattern recognition.

The first DCT coefficient of each frame represents the energy of that feature. This energy term has the largest value but carries relatively less information. Therefore, the energy term may weight too much in the computation, since an unweighted Euclidean distance measurement is used. To examine this possibility, we did another set of tests using DCT coefficients 2-6 and 2-8 as feature components. With DCTC's 2 to 6 as the input parameters, test results improved 8% and training results improved about 1%. The best recognition rate of 72.26% was obtained with 64 codewords. If more parameters were used, the recognition dropped as shown in table 2-4.

In all cases, performance deteriorates with an increasing

vocabulary size. For example, a 93.90% testing recognition rate was obtained with a 9-word data base, whereas only 44.01% testing recognition rate was achieved with all 99 words. However, for training, 100% recognition rate can be obtained for both 9 words and 99 words. This may be due to the insufficient training data processed. If we can increase the amount of data for training, we can expect that the training results will decrease and the testing results will increase which is shown in table 2-3.

After performing all the tests reported in this chapter (plus many others not reported here), we were confident that the HMM implementation was correct. We also optimized some aspects of the model for our particular data base and obtained valuable control results. Therefore the tools were available for the primary experiments.

99 CVC WORD RECOGNITION				
	REGULAR (10 dim. feature space)		REGULAR (10 dim. feature space)	
CODEWORD	TRAIN(%)	TEST(%)	TRAIN(%)	TEST(%)
2	5.21	4.16	4.93	2.88
4	13.57	6.43	14.20	6.79
8	26.73	12.46	31.21	15.22
16	50.45	24.64	56.74	27.07
32	72.93	30.32	75.57	33.04
64	91.02	38.19	91.92	39.82
128	98.63	43.26	98.56	42.22
256	99.93	44.01	99.95	44.32

Table 2-1. Effect of switching the training speakers (15) with the testing speakers (15) on recognition rate.

	30 CVC WORD RECOGNITION			
	REGULAR (without the transitions that skip states)		REGULAR (with the transitions that skip states)	
CODEWORD	TRAIN(%)	TEST(%)	TRAIN(%)	TEST(%)
2	10.76	8.56	8.07	8.90
4	23.53	18.49	22.35	14.38
8	34.11	26.37	41.34	27.40
16	71.26	43.84	70.25	42.12
32	86.72	57.19	85.88	55.14
64	95.17	64.38	95.46	59.93
128	97.82	61.30	97.82	60.96
256	100.0	63.36	100.0	62.67

Table 2-2. Effect of different types of hidden Markov models on recognition rate. (20 speakers for training and 10 speakers for testing)

	30 CVC WORD RECOGNITION			
	REGULAR (20 spk. for training 10 spk. for testing)		REGULAR (15 spk. for training 15 spk. for testing)	
CODEWORD	TRAIN(%)	TEST(%)	TRAIN(%)	TEST(%)
2	10.76	8.56	9.44	9.05
4	23.53	18.49	27.87	14.93
8	34.11	26.37	45.39	29.41
16	71.26	43.84	72.13	41.63
32	86.72	57.19	89.89	52.26
64	95.17	64.38	97.53	61.31
128	97.82	61.30	99.78	57.47
256	100.0	63.36	100.0	60.63

Table 2-3. Effect of data base size for training and testing on recognition rate.

	30 CVC WORD RECOGNITION			
	REGULAR(1) (5 dim. feature space parameter 2 -> 6)		REGULAR(1) (7 dim. feature space parameter 2 -> 8)	
CODEWORD	TRAIN(%)	TEST(%)	TRAIN(%)	TEST(%)
2	24.71	18.15	24.37	18.49
4	53.11	39.04	53.78	43.84
8	72.27	57.53	73.28	50.34
16	84.87	62.67	84.87	63.36
32	93.11	71.58	92.27	71.58
64	96.64	72.26	96.81	68.15
128	99.33	70.89	99.50	68.49
256	99.83	67.47	99.83	68.49

Table 2-4. Effect of feature vector selection on recognition rate. (20 speakers for training and 10 speakers for testing)

2.5.3 Summary

In this chapter, we showed that the techniques of vector quantization of DCT vectors and hidden Markov modeling can be combined in a simple, straightforward manner to implement a speaker-independent, isolated-word recognizer. We first defined the hidden Markov model and the methods we used to implement the model. We then described some of the vector quantization techniques and the codebook implementation techniques. Verification results for a variety of test conditions were presented in the last part of this chapter. These tests indicated that the hidden Markov model we implemented functions properly.

CHAPTER 3

PREPROCESSING FOR THE HIDDEN MARKOV MODEL SPEECH RECOGNITION

3.1 Overview

The problems encountered in practical speech recognition can be summarized as: (1) Talker variations: No two people sound alike. (2) Ambiguity: Acoustical variables are not mapped one to one onto phonemic variables. (3) Variations in individual speech: These include variations due to carelessness, coarticulation, and temporal changes for repetitions of the same word. (4) Noise and interference: Noise degrades the performance of any ASR system.

There are three major problems in using raw spectral feature parameters for ASR with a discrete HMM classifier. First, phonetic differences do not necessarily correspond to Euclidean distances in the feature space. Since the vector quantization step of a discrete HMM uses Euclidean distances in developing the codebook, there is a poor match between codewords and phones. This discrepancy can cause confusion in the overall recognition system. Second, phonetic identity depends on information distributed across several speech frames. Therefore, as for the first point listed, using codewords developed from single frames, there will again be a poor match between codewords and phones. Third, since the

original features encode not only phonetic variations but also many other sources of variation in the speech signal, the number of features required is large. Although an HMM can still be trained using these features, the amount of speech training data required may be unmanageably large. Stated another way, with a fixed amount of speech materials, presumably better overall performance will be obtained if the features used as the input to the HMM primarily reflect the phonetic differences needed to distinguish the words in the vocabulary.

Despite the above problems, with any "reasonable" set of spectral parameters, HMM's are able to easily distinguish words that differ in several phones and thus sound significantly different. However, for vocabularies that contain similar sounding words (for example, words differing in only one phoneme), recognition performance can be poor. In this thesis, we introduce techniques to enhance the phonetic differences among words, so that phonetically-similar words can be more easily recognized. For these techniques, a transformation is used to project data to either the same or a reduced-dimensionality subspace with the goal of enhancing phonetic information in the data while discarding the non-phonetic variations. Since phonetic information is in general distributed across several speech frames, the transformations project several initial speech frames (a block) to each transformed frame. The techniques explored in this thesis include linear discriminant analysis (LDA) and a

discrete cosine transform (DCT). In both cases, these techniques were used to compute a "sliding" block transform preprocessor for use with a HMM speech recognition system.

By using preprocessing, we can not only enhance the phonetic differences between the words, but also potentially reduce the dimensionality of the feature space. With a finite amount of speech materials, a lower dimensionality space may enable better HMM training and thus improve performance. However, data reduction may improve or degrade overall ASR system performance, depending on the details of the processing. If too much phonetic information is lost through data reduction, the advantage of the preprocessing may not overcome the information lost. The overall effects of the transformations can only be evaluated experimentally.

3.2 System Definition

The overall isolated-word ASR system is shown in figure 3-1. It consists of front-end feature extraction, transformation of spectral features to phonetic features, and a HMM classifier. The primary focus of this study was the linear transformation block shown in figure 3-1. In this section we give a brief explanation of the initial feature extraction and summarize the parameters used for the VQ and HMM stages (Recall a more detailed explanation of VQ and HMM was given in chapter two). We also explain the operation of the linear transformation block and describe the two methods used to compute the linear transformation coefficients.

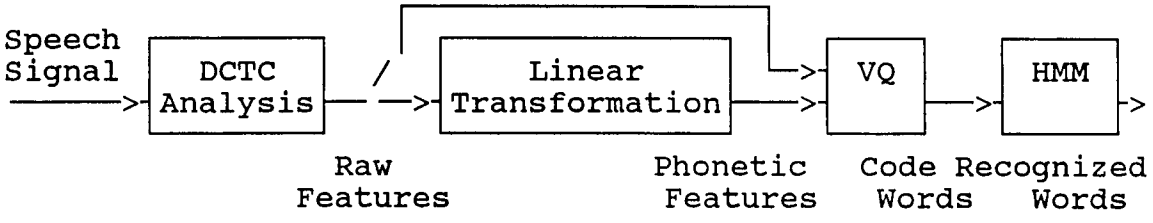


Figure 3-1. HMM based recognition system with different transformations as front-end.

3.3 Feature Parameter Extraction

The speech signal was first preprocessed as described in chapter two. The speech spectral features selected were discrete cosine transform Coefficients (DCTC), which are the a_n's in the equation

$$H'(f') = \sum_{n=1}^{n=M} a_n \cos((n-1)*\pi*f') \quad . \qquad (3\text{-}1)$$

The H' implies nonlinear amplitude scaling of the magnitude spectra, and f' implies a nonlinear frequency scaling. The frequency scale is normalized so that a selected frequency range in f of $f_1 \leq f \leq f_2$ corresponds to $0 \leq f' \leq 1$. The a_n's are computed after first high-frequency preemphasizing the signal $(1-.95\ z^{-1})$, using a 20 msec Hamming window, and computing the magnitude spectra for each frame (1 DFT and 1 IDFT). We used log amplitude scaling, a bilinear frequency warping with a coefficient of .5, and a frequency range of 150 Hz to 6000 Hz here, and the first 10 coefficients as spectral features. The frame spacing was 10 ms. These spectral features, and these particular values for frame length, frequency range, etc., were chosen based on previous experiments conducted in the speech laboratory at Old Dominion University (Zahorian and Gordy, 1983; Shen, 1988; Nossair, 1989).

1. VQ and HMM Processing

The vector quantizer and HMM were implemented using the methods described in chapter two. For the VQ step, the number

of codewords was varied from 2 to 256. The HMM was implemented with five-state, a left-to-right model with only self-loops and transitions to the next state allowed.

2. Linear Transformation

The linear transformation block of figure 3-1, as mentioned above, was the main focus of this investigation. This transformation operated as follows. N frames, each with M features per frame, were matrix multiplied with a P by (N * M) matrix to obtain P transformed data points per block. This process was repeated for every L original frames. Thus the original feature space was transformed by a sliding block linear transform operation. Two primary methods were investigated for computing the block transformation coefficients, as discussed later in this chapter. Variables for each method included the number of frames per block and the spacing between blocks. The objective of this transformation was to enhance phonetic contrasts. Therefore the outputs of the linear transformation are referred to as phonetic features.

Note that there is no fundamental reason for restricting the transformation to the phonetic feature space to a linear transformation. In fact, a nonlinear transformation based on a neural network might yield superior results to the linear transformation. However, time constraints prevented the completion of experiments with a neural network processing step.

3.4 Discrete Cosine Transform Linear Transformation

3.4.1 Introduction

Each speech frame contains only static information based on the time instant for that frame. However as mentioned previously, phonetic information is distributed over several speech frames. Therefore, a HMM recognition system based on single frames may not perform well due to the poor correspondence between VQ codewords and phonemes. A better approach is to use several successive frames jointly to develop codewords which would then represent speech information distributed over longer time intervals. One way to combine several frames into a block is to smooth the time trajectory of each parameter using a discrete cosine transformation over time for each parameter. Each block of frames can then not only capture the static spectral information, but also can incorporate dynamic information, i.e. information about the rate of change of the spectrum over time. It is expected that this transform will not only enhance the speech information contained in each block, given the spacing between blocks is sufficiently large, but also can reduce the amount of data needed for computation.

A discrete cosine transformation of parameter time trajectories has been used for phoneme recognition to reduce the amount of data in previous experiments (Shen, 1988) with a Bayesian Maximum Likelihood Classifier. Such a method is also quite similar to other methods which have been reported in the literature (Hanson and Applebaum, 1989) for enhancing

HMM recognition. However the actual effectiveness of this transform, in terms of transform length, number of terms, amount of overlap, etc. can only be evaluated through experimental testing.

3.4.2 Algorithm

A three-term cosine basis vector expansion was chosen for our experiment. This technique can be explained as follows. The parameters are arranged as a two-dimensional array (matrix) with the column index corresponding to time and the row index corresponding to the parameter index. For example, with 10 parameters per frame and 12 frames, this matrix is 10 x 12. Each row of the matrix is thus the time trajectory of one parameter.

Each row s(n), $1 \le n \le N$, may be expressed in a three-term cosine expansion as follows.

$$A_0 = (1/N) \sum_{n=1}^{N} s(n) \quad , \tag{3-2}$$

$$A_1 = (1/N) \sum_{n=1}^{N} s(n)*\cos(\pi(n-.5)/N) \quad , \tag{3-3}$$

$$A_2 = (1/N) \sum_{n=1}^{N} s(n)*\cos(2\pi(n-.5)/N) \quad . \tag{3-4}$$

Therefore,

A_0 is the average value over all the samples;

A_1 is the coefficient of the basis vector consisting of

a half-cycle of a cosine and is thus a measure of the rate of change of s(n) over the block; and,

A_2 is the coefficient of the basis vector consisting of a full cycle of a cosine over and thus gives more detailed information about the trajectory of s(n) over the block.

In our study, we segmented each stimulus into several blocks. The number of frames per block was either specified as a fixed length or adjusted according to an error criteria as explained below. In each case, the data was represented by the A0, A1, A2 coefficients for each parameter. Thus the number of features required to encode each block was 3 times number of original parameters per frame. Assuming for the moment no overlap of blocks, this procedure results in data reduction if the block length is greater than three frames. In general, with this approach, the number of parameters per frame increases, (from 10 to 30) but the number of frames decreases.

For the case of variable block lengths based on error the following approach was used.

First, the trajectory over the block for each parameter could be represented exactly by an N term cosine expansion as:

$$s_i(n) = \sum_{k=1}^{N} a_{ik} * \Phi_k(n), \qquad \text{for } 1 \le i \le M,\ 1 \le k \le N, \qquad (3\text{-}5)$$

where $\Phi_k(n) = \cos(\pi*(k-1)*n/N), \quad 1 \le n \le N.$

However, since only three terms in the expansion were used, each parameter was in effect approximated by

$$s'_i(n) = \sum_{k=1}^{3} a_{ik} * \Phi_k(n). \qquad (3\text{-}6)$$

From the theory of orthogonal basis function signal representations, the normalized mean square difference between s and s', summed over all s_i's is given by

$$E = \sum_{i=1}^{M} \sum_{k=4}^{N} a_{ik}^2 \quad / \quad \text{total energy} \quad , \qquad (3\text{-}7)$$

$$\text{where total energy} = \sum_{i=1}^{M} \sum_{k=1}^{N} a_{ik}^2 \quad .$$

Thus E is the normalized mean square error which would result from approximating each s_i by s_i'.

For the variable block length approach, the block length was allowed to increase until E reached 0.001. Thus the block length was a function of the original data. The block length is longer if the spectra is slowly changing (i.e., vowels), and shorter if the spectra is rapidly changing (i.e., stops). This approach time-normalizes the original data to a certain extent.

3.4.3 Experiments and Results

All the experimental results are presented in table 3-1. All results were obtained using the first 30 words listed in Appendix A. The recognition results are much higher for

training versus testing, indicating insufficient training data. Results are given for the two basic methods for implementation mentioned above--data independent block length and block length dependent on error. For the data-independent block length case, we investigated the effects of block length and amount of block overlap on the recognition rate. The best test performance obtained was 51.71% with 256 codewords, a frame length of four, and block spacing of three. The performance is degraded when the block length is increased. The results change by only a small amount (less than 1%) if the frame spacing is increased. However, all of these test recognition results are lower than the control results given in chapter two.

Results for the data-dependent block length are also given in table 3-1. The results given were obtained by setting the percentage error to 0.1% with one frame overlap. These conditions resulted in a data reduction of approximately 50%. The recognition rates were even lower than for the fixed block length. In other experiments with a reduced percentage error used to compute the block length, the data size increase rapidly, but not much improvement in recognition accuracy was achieved.

In summary, for all conditions tested with the discrete cosine transform, the overall performance was worse than for the original features. Even for the best case, test recognition results were 10% lower than for the original data. One of the apparent reasons for the failure of this method was

that after we projected the ten dimensional parameter vectors onto reduced dimensionality feature spaces, useful acoustic information was discarded. We also suspected that there was not enough training data to take advantages of the increased potential accuracy provided by the discrete cosine transformation. That is each block (which takes on the role of a frame in the transformed data) contains much more information than a single frame, which thus requires more codewords and thus more training data. The results might change with a vastly increased amount of training data. The third reason for poor results is that discrete cosine transformation treats all the phonemes the same. It does not emphasize phonetic distinctions. This approach appears to be more of a data reduction technique than a phonetic enhancement technique. We found no evidence indicating that a data-dependent block length is superior to a fixed block length.

		CODEWORD					
		64		128		256	
		TRN(%)	TST(%)	TRN(%)	TST(%)	TRN(%)	TST(%)
REGULAR (30 words)		95.17	64.38	97.82	61.30	100.0	63.36
3 TERM COSINE EXPANSION	Adjustable frame length based on percentage error of .1% (1 frame overlap)	84.25	29.11	95.97	35.62	99.12	35.96
	1 frame overlap. 8 frm./blk.	81.41	39.73	94.87	39.73	98.58	39.04
	1 frame overlap. 4 frm./blk.	89.20	48.28	95.93	52.74	98.94	51.71

Table 3-1. Effect of dynamic information (Three-term cosine expansion) on recognition rates for different conditions.
(20 speakers for training and
10 speakers for testing)

3.5 Linear Discriminant Analysis (LDA) Transformation

3.5.1 Introduction

Discriminant analysis is a linear transformation that maps an element (usually a vector) from the original space to an element in a reduced-dimensionality subspace such that predefined groups are well-clustered in the transformed space. This technique was originally developed by Ronald Fisher in 1923. The transformation coefficients are computed from the between-class covariance matrix and within-class covariance matrix of the data. After the discriminant analysis, the ratio of between-class variance to the within-class variance is maximized. Therefore the data points in each category will be well-clustered according to the specified groups in the reduced-dimensionality subspace.

In our experiments, discriminant analysis was used mainly with classes defined in terms of phones, phone segments, and diphones. The objective was to transform the raw feature space such that features for the same phone would have similar values and feature values for different phones would be different. Thus phonologically equivalent sounds should be well clustered in the feature space and phonologically distinct sounds should be well separated. Furthermore, Euclidean distance in the transformed space should be a good measure of phonetic distance. Another advantage of this technique is that, if the transform is computed from a block of several frames of features, both static and dynamic information can be used to compute the transformed features.

The basic assumptions underlying linear discriminant analysis include multivariate normal distributions and equal covariance matrices for all groups. Violations of these assumptions result in a suboptimal solution for the transformation. The dimensionality of the transformed space obtained from discriminant analysis is restricted to the minimum of the original number of variables or one less than the number of data categories.

3.5.2 Theorem and Algorithm

The key for the LDA computation is to find a transform matrix U which can map an original P-dimensioned parameter vector $X=[x_1,\ldots,x_p]$ to a final N-dimensioned vector $F=[f_1,\ldots,f_N]$. Therefore the dimension for the U matrix is N x P. Mathematically, the relationship among these matrices are as follows:

$$F^t = U\ X^t\ . \qquad (3\text{-}8)$$

Three different matrices must first be computed before the transform matrix U can be obtained. These three matrices are as follows.

1. Grand covariance matrix T.

Assuming that the M training vectors under investigation comprise G distinct groups, then an arbitrary entry t_{ij}, of row i and column j of T, is expressed as

$$t_{ij} = \sum_{k=1}^{M} (X_{ki} - X_{\cdot i})(X_{kj} - X_{\cdot j}) \quad , \tag{3-9}$$

where $1 \le i \le p$, $1 \le j \le p$,

X_{ki} = the value of variable i of the k^{th} parameter vector, and

$X_{\cdot i}$ = grand mean value of variable i over all the parameter vectors.

2. Within-groups covariance matrix W.

This matrix is very similar to T, except that the deviations are measured from the mean of the group to which the case belongs. For a specific training group R, with S parameter vectors, the elements of W are defined as:

$$W_{ij} = \sum_{s=1}^{S} (X_{si} - X_i)(X_{sj} - X_j), \tag{3-10}$$

where $1 \le i \le p$, $1 \le j \le p$,

X_{si} = the value of variable i for the s^{th} parameter vector in group R, and

X_i = mean value of variable i for all parameter vector in group R.

3. Between-groups covariance matrix.

This matrix is the difference between the grand covariance matrix and the within-group covariance matrix.

$$B = T - \sum_{i=1}^{G} W_i = T - D, \quad \text{and} \quad D = \sum_{i=1}^{G} W_i \ , \tag{3-11}$$

where W_i is the covariance matrix of the i^{th} group.

It is safe to assume that D is nonsingular, and hence possesses the inverse D^{-1}. The rows of the transform matrix U are the eigenvectors of the matrix $D^{-1}B$. A more-detailed derivation of discriminant analysis is given in Cooley and Lohnes (1971).

The discriminant model may be interpreted as a special type of factor analysis that extracts orthogonal factors of the measurement battery for the specific task of displaying and capitalizing upon differences among criterion groups. The model derives the components which best separate the cells or groups of a taxonomy in the measurement space. The possible rank n of the discriminant subspace to be fitted in the measurement space depends on the relative sizes of g (the number of groups), and p (the number of elements in the vector variable). Their relations are n = min (g-1 , p). For example, the centroids of two groups have to coexist on a single line, regardless of the number of variates in each centroid. The centroids of three groups have to coexist in a plane. The best discriminant plane has the attraction that graphic displays of the locations of the groups in it can easily be presented.

3.5.3. Use of LDA to Enhance Phonetic Distinctions in the Transformed Space

Brown (1987) has proposed using several successive frames jointly in order to model the joint density of the observed

speech more accurately. He then used linear discriminant analysis to reduce the number of dimensions. Hunt et. al. (1988, 1990) also used LDA to improve ASR results for isolated-word recognition.

In our study LDA was used with the goal of improving phonetic discriminability in the transformed space. Since the classes were defined as phones (or variants of phones as described below) the application of LDA required training data segmented according to phone boundaries. The phonetic segment boundaries, used to mark the data for computing the within and between class matrices required by LDA, must be reasonably accurate in order for the linear transform to really enhance the phonetic distinctions. The CVC data base, described previously, was carefully hand labeled according to acoustic segments in each phone of each word. Thus such data was well-suited for use with the LDA phonetic-category transformation.

In applying LDA to enhance phonetic discriminability in a feature space for use with HMM ASR, there are several basic issues to investigate as follows.

1. The Definition of the Phonetic Classes.

The most obvious method for defining phonetic classes is simply to group all data for all frames of each phone as comprising that phone. Thus, for example, all frames of an initial /b/ for all CVC's in the training data would be grouped with all frames of final /b/'s for all the CVC's in the data base. With this method the number of phonetic classes is simply equal to the number of distinct phones in

the data. However, this method for defining classes may not perform well since the acoustic properties of (for example) final stops would be expected to be quite different from the acoustic properties of initial stops. Several methods for defining the number of phonetic classes and the selection of the data for defining that class were investigated, as discussed in the section on experimental results.

2. The Number of Frames Per Block.

The number of frames per block used for the LDA could range from one to any number desired. That is each frame of transformed data would be computed from all the frames of that block. A larger block size would provide more dynamic information. However too large a block size might result in too many variables to allow stable estimates of LDA basis vectors for a finite amount of training data.

3. The Number of LDA Basis Vectors.

As mentioned previously, the number of LDA basis vectors is limited to the minimum of the number of original variables or one less than the number of groups. For our experiments, the number of variables ranged from 10 (10 parameters per frame, 1 frame per block) to 100 (10 parameters per frame, 10 frames per block). The number of phonetic groups ranged from 18 to over 80. Nevertheless, although we could have generally used a large number of LDA basis vectors, we used only five or ten basis vectors in our experiments since most of the discrimination power of the LDA transformed space is contained in the first few dimensions.

3.5.4 Experiments and Results

We computed the within (phoneme) class and between class means and covariances of selected frames of the training data. We used the generalized eigenvector solutions to find the best set of linear discriminant features. The objectives of the experiments were to determine if LDA preprocessing can be used to improve HMM isolated-word recognition and to systematically investigate the issues listed above for optimizing the application of the LDA preprocessing technique. Not all combinations of methods could be explored because there were simply too many. Therefore we attempted to determine the best value of each variable individually (such as block length) before investigating the effects of other variables. The primary series of experiments was conducted with the first 30 words listed in Appendix A. Additional verification experiments were conducted using 50 words for final recognition and 49 words for computing the LDA matrix.

3.5.4.1 Phonetic Class Effects and LDA Dimensionality Effects

Experiment Set One.

The goals of the first set of experiments were: (1) to determine if LDA preprocessing appears to have any promise for improving HMM speech recognition; (2) to determine if recognition rates are improved if initial and final stops are considered as the same phonetic category (for the same phone)

or separate phonetic categories; and, (3) to determine the effects of the dimensionality of the transformed space (five versus ten dimensions). This experiment was performed with the 30 word data base. One frame per block and a frame spacing of one was used throughout. Twenty speakers were used for computing the LDA transform matrix and for training the HMM's. The other ten speakers were used for testing (Note that the LDA transform was applied to both the training and test speakers). In all cases, the initial stop was defined as the beginning of each word up to the midpoint of the IT segment, the vowel was considered the portion of the word from the midpoint of the IT to the midpoint of the FT segment, and the final stop was considered the portion from the midpoint of the FT segment to the end of the word. All frames belonging to each phone were used in computing the within-class covariance matrix for that phone. For one case (A), initial and final stops were grouped. For the other case (B), initial and final stops were considered as separate phonemes. For the grouped case there were 18 phonetic classes. For the separated case there were 24 phonetic classes.

The results of the experiment are summarized in table 3-2. The table gives results for the two basic cases (A and B), for five and ten-dimensional transformed spaces, and for various numbers of codewords used with the HMM. Examination of the table, as well as a comparison with control results from table 2-2, indicates the following points. First, with a small codebook size, the test recognition results increase

dramatically over the control results. For example with 16 codewords the test result for case B is 64.38% versus 43.83% from table 2-1. Second, with a larger number of codewords, the LDA processing does improve the results over the control case, but the percentage change is smaller than for the smaller number of codewords. For example, with 128 codewords the test result for case B is 67.47% versus 61.30% from table 2-2. Third, a ten-dimensional transformed space is superior to a five-dimensional transformed space in terms of automatic recognition results. Fourth, case B (initial and final stops considered as separate phonetic categories) is significantly better than case A (initial and final stops considered as the same category).

The general conclusion of this experiment is that LDA processing does show some promise for improving HMM isolated-word recognition. The details of the use of LDA does effect the results. Additional experiments, described in subsequent sections, were used to help improve the implementation details.

30 CVC WORD RECOGNITION				
	DISCRIMINANT (case A)		DISCRIMINANT (case B)	
	5 dim. feature space. Initial & final stops are grouped. (case A.1)		5 dim. feature space. Initial & final stops are separated.(case B.1)	
CODWORD	TRAIN(%)	TEST(%)	TRAIN(%)	TEST(%)
32	89.41	61.99	90.76	64.38
64	93.95	63.36	94.12	65.07
128	98.32	59.25	98.48	65.07
256	99.83	59.59	99.66	61.99
	10 dim. feature space. Initial & final stops are grouped. (case A.2)		10 dim. feature space Initial * final stops are separated.(case B.2)	
CODWORD	TRAIN(%)	TEST(%)	TRAIN(%)	TEST(%)
32	91.26	65.06	90.93	67.12
64	96.81	68.49	95.97	70.89
128	99.70	65.15	99.16	67.47
256	100.0	66.90	100.0	69.17

Table 3-2. Effect of initial and final stops grouping and feature dimensionality on recognition rate. (1 frm./blk. 1 frm. space. Transform matrix based on training speakers. 20 speakers for training and 10 speakers for testing.)

Notes:

1. Case A --Initial and final stops considered as the same category.
2. Case B --Initial and final stops considered as separate phonetic categories.
3. For case A.1 with 2,4,8,16 codewords, Training results are 14.62%, 48.74%, 72.44%, and 79.5%, Testing results are 12.33%, 37.33%, 55.82%, and 61.30%.

3.5.4.2 Block Length Effects

Experiment Set Two

The objective of the second group of experiments was to determine the effect of the number of frames per block on recognition performance. These experiments were performed only for the initial and final stops considered as separate phonemes, using the results from the first set of experiments. The experimental conditions were identical to those for case B of Experiment one, with ten parameters per transformed frame, except that five frames per block were used as the input for the LDA processing. Results are given in table 3-3. For ease of comparison, the results from experiment one for the identical conditions except for number of frames per block, are also repeated in the table. The results show that in general there is a small improvement in recognition rate with five frames per block verse one frame per block. Presumably the transform based on five frames per block does make use of some dynamic information in the speech signal.

Since the results for five frames per block were somewhat better than for one frame per block, an additional test was made with ten frames per block for the case of 128 codewords. The test result was, however, slightly worse (2%) than the result for five frames per block. Therefore, of the conditions tested, we concluded that five frames per block should be used for further experimentations. One additional test was made with five frames per block to examine tradeoffs

between number of transformed parameters per frame and frame spacing. In particular, the LDA processing was set up to yield 16 transformed parameters per frame but with a frame spacing of two frames. Thus the total speech data was nearly equivalent to ten parameters per frame and a frame spacing of one. This data arrangement was tested only for 128 and 256 codewords. The results shown in table 3-4 indicate an improvement in performance.

30 CVC WORD RECOGNITION				
	DISCRIMINANT 1 frm./blk. 1 frm.space		DISCRIMINANT 5 frm./blk. 1 frm space	
CODEWORD	TRAIN(%)	TEST(%)	TRAIN(%)	TEST(%)
2	18.99	15.75	18.32	15.75
4	41.68	37.33	39.50	33.56
8	74.96	60.62	71.43	57.19
16	82.52	64.38	83.53	66.43
32	90.93	67.12	86.72	68.84
64	95.97	70.89	93.78	70.55
128	99.16	67.47	97.98	69.86
256	100.0	69.17	99.50	72.26

Table 3-3. Comparison of results as a function of block length.
(10 dim. feature space.
Initial & final stops are separated.
Transform data computed from 30 words.
Transform matrix based on training speakers.
20 speakers for training and
10 speakers for testing.
The maximum number of clusters is 24.)

30 CVC WORD RECOGNITION				
	DISCRIMINANT 5 frm./blk. 1 frm.space		DISCRIMINANT 5 frm/blk. 2 frm.space	
CODEWORD	TRAIN(%)	TEST(%)	TRAIN(%)	TEST(%)
128	97.82	75.0	97.31	75.34
256	98.82	75.0	99.16	75.69

Table 3-4. Effect of frame length and frame spacing on recognition rate.
(16 dim. feature space.
Transform matrix based on training speakers.
Initial and final stops separated.
20 speakers for training and
10 speakers for testing.)

3.5.4.3 Acoustic Region Effects

Experiment Set Three.

Since the results of the first experiment indicate that better performance is achieved if initial stops and final stops are considered as separate phonetic classes, we investigated whether or not better results could be obtained if even "finer" distinctions were used to define phonetic classes. These regions were defined in terms of a combination of acoustic labels (IB, IT, SV, FT, FB) and phoneme/diphone (V, C, CV, VC) labels. In particular six "acoustic/phonetic" regions were defined as shown in figure 3-2. An "equivalence" class was formed for all data having the same acoustic label and the same phonetic label. The starting points for the six acoustic regions were defined as:

(1) the beginning of IB,

(2) the beginning of IT,

(3) one frames before the beginning of SV,

(4) three frames before the end of SV,

(5) five frames before the end of FT, and

(6) the beginning of FB.

Region (1) was considered a consonant region. Thus data in region (1) with the same consonant label would be comprise one phonetic class for LDA. Region (2) was considered a CV region. Thus data in region (2) with the same initial consonant label and the same vowel label would be considered as comprising one phonetic class. Similarly regions (3) and (4) were considered as two vowel regions (beginning and

ending), region (5) was considered a VC region, and region (6) was considered as a final stop region. The amount of data used from each region for covariance matrix computations was variable in terms of number of frames per block, frame spacing, and total number of blocks. Thus it was possible that some of these regions might overlap. For most of the experiments conducted only one or five-frame block were used from each region of each word for covariance matrix computations.

The first set of experimental results based on five of the phonetic regions (all except region (5)) described above are given in table 3-5. This experiment was run with five frames per block and one block for each region for covariance matrix computations. Note that the test recognition results, for the cases of 128 and 256 codewords, are much higher than for the best results obtained from Experiment two (80.14% versus 72.26%). However inspection of the LDA program, after the results were obtained, indicated that an error had been made in the timing of the start points for the data in half of the words. That is, for half of the words, the phonetic regions did not properly line up with the acoustic segment labels for those words. In effect noise had been introduced in generating the timing signal for the acoustic labels for half of the words.

Assuming that even better results would be obtained with precise timing information for all words, the program error was eliminated and another series of related experiments were

conducted. The objective was to examine the effects of the number of regions used to define phonetic classes. The number of regions was varied from four to six, using subsets of the regions defined above as listed in the table 3-6. In general, the results are quite similar as the number of regions changes from four to six. The absolute best result of 73.63% was obtained with 64 codewords and four segments. On the average, however, for the four codebooks evaluated, the results are the best for five regions. Unfortunately, even this "best" result is worse than the result given in table 3-5, obtained with the timing error.

Recognition results are also plotted in figure 3-3, as a function of the number of acoustic regions used to define phonetic classes and as a function of how data was selected from each region to compute the covariance matrices for LDA. The details of the conditions are listed in the figure 3-3 caption. For one condition (#8), random noise was added to the timing information use to define the acoustic segments. The data is plotted only for the case of 128 codewords. Note that in general test results vary by only a small amount from the worst to the best condition plotted in figure 3-3. However, the condition with the best results is #3 (72.96%) corresponding to five segments and five frames per block.

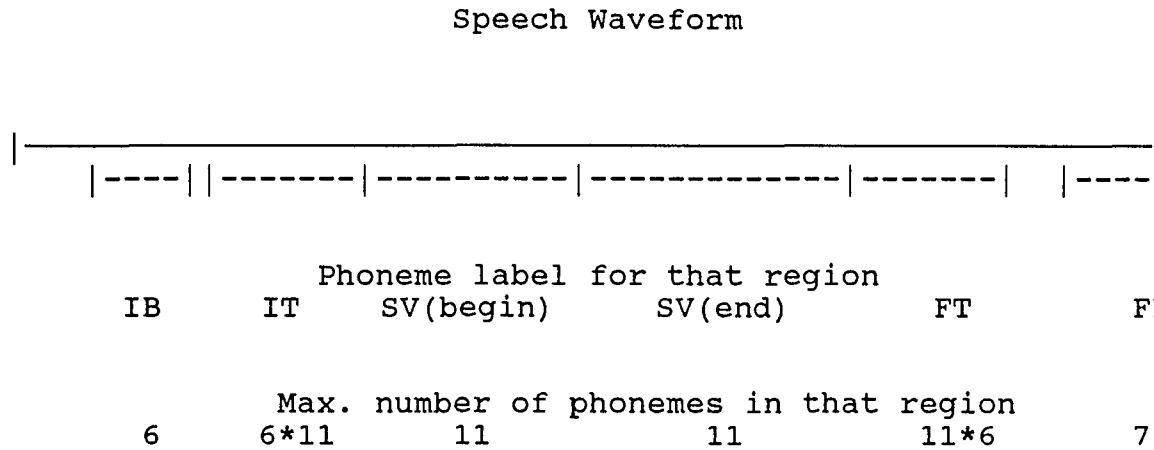


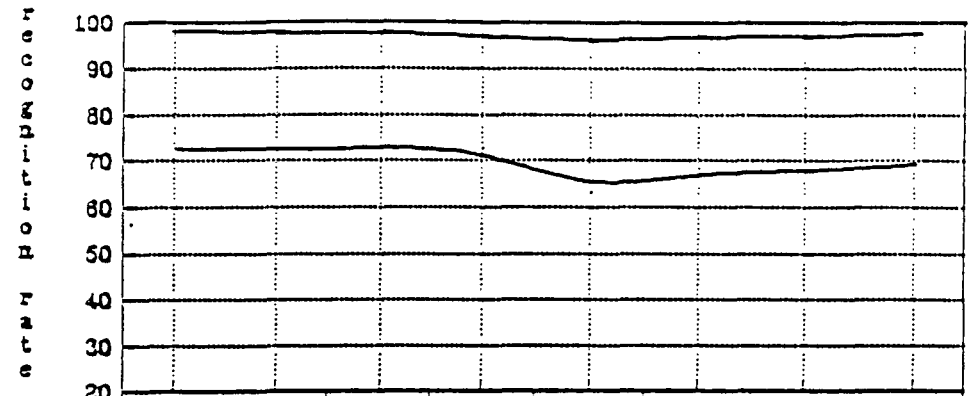
Figure 3-2. Six region segmentation of the speech waveform. Boundary is determined by using the phonetic segmentation information. (some segments may be overlapped)

30 CVC WORD RECOGNITION		
	DISCRIMINANT (10 dim. feature space 5 regions)	
CODEWORD	TRAIN(%)	TEST(%)
32	90.59	69.18
64	95.29	68.49
128	98.32	78.42
256	99.50	80.14

Table 3-5. Effect of dividing the waveform into five regions on the recognition rate. (with some mistakes in data selection) (Transform matrix is based on training speakers 20 speakers for training and 10 speakers for testing.)

30 CVC WORD RECOGNITION						
	DISCRIMINANT (4 regions, 30 clusters)		DISCRIMINANT (5 regions, 60 clusters)		DISCRIMINANT (6 regions, 86 clusters)	
CODEWORD	TRN(%)	TST(%)	TRN(%)	TST(%)	TRN(%)	TST(%)
32	90.42	67.12	90.25	69.86	89.08	69.52
64	95.63	73.63	94.79	71.23	94.45	71.23
128	97.65	72.26	97.82	72.96	94.17	72.60
256	99.66	70.55	99.66	72.94	99.33	68.15

Table 3-6. Effect of dividing the waveform into different regions on the recognition rate. (10 dim. feature space. 5 frm./blk. and 1 frm. space. Transform matrix is based on 30 speakers. 20 speakers for training and 10 speakers for testing.)



	1	2	3	4	5	6	7	8
TEST	72.6	72.26	72.96	72.6	63.01	68.15	67.46	69.17
TRAIN	98.32	97.65	97.82	97.14	95.97	97.14	96.97	97.65

Figure 3-3. Effect of number of segments on recognition rate. (128 codewords)

1 ---- 3 segments, IB,SV,FB, 21 clusters.
2 ---- 4 segments, IB,SVb,SVe,FB, 30 clusters.
3 ---- 5 segments, IB,IT,SVb,SVe,FB, 60 clusters.
4 ---- 6 segments, IB,IT,SVb,SVe,FT,FB, 86 clusters.
cases 1,2,3,4 are tested under 5 frm./blk. 1 frame space (1,10,5,1,1)

5 ---- 5 segments, 5 frm./blk. 2 frame space. (1,10,5,2,2)
6 ---- 5 segments, 1 frm./blk. 1 frame space. (1,10,1,1,1)
7 ---- 5 segments, 10 frm./blk.1 frame space. (1,10,10,1,1)
8 ---- 5 segments, with random noise in ± 3 frames (1,10,5,1,1,6)

3.5.3.4 Verification with Large Vocabulary

Experiment Set 4

The objective the last series of tests was to verify the results from the first three sets of experiments using a larger number of words for testing. In particular the entire set of 99 words was divided into one group of 50 words and another group of 49 words. These two groups were approximately phonetically balanced, with each group containing all the vowels and consonants in the 99 word data base. One group of words was used for LDA covariance matrix computations (using all 30 speakers). The LDA transform computed from one set was then applied to the other set of words. These words were processed by the VQ/HMM system, with and without the LDA preprcessing. In all cases 15 speakers were used for training HMM's and 15 were used for testing.

The results of the experiments are given in table 3-7. The table shows that LDA preprocessing does improve recognition results about 10% over no preprcessing. As a final verification, the HMM's were computed with DCTC's 2-6 as raw features (table 3-8). This test was made because of the relatively good results obtained with that particular feature set for the 30 word tests. However, as table 3-8 shows, DCTC's 2-6 give results only slightly better than DCTC's 1-10 for the 50 word vocabulary. Thus the LDA preprocessing did significantly improve the results over any conditions of raw parameters tested for the 50 word vocabulary.

99 CVC WORD RECOGNITION				
	REGULAR DATA		DISCRIMINATED DATA	
	49 words for VQ/HMM		50 words for LDA and 49 words for VQ/HMM.	
CODEWORD	TRAIN(%)	TEST(%)	TRAIN(%)	TEST(%)
32	85.54	42.52	87.33	55.26
64	93.94	49.86	95.59	59.28
128	99.45	54.99	98.77	63.43
256	99.86	55.26	99.86	62.74
	50 words for VQ/HMM		49 words for LDA and 50 words for VQ/HMM.	
CODEWORD	TRAIN(%)	TEST(%)	TRAIN(%)	TEST(%)
32	91.26	55.06	90.93	67.12
64	96.81	58.49	95.97	70.89
128	99.70	55.15	99.16	67.47
256	100.0	56.90	100.0	69.17

Table 3-7. Effect of using different set of data for LDA and VQ/HMM on recognition rate. (5 regions, 5 frm./blk., and 1 frm. space. Transform matrix based on 30 speakers 15 speakers for training and 15 speakers for testing.)

50 CVC WORD RECOGNITION		
	DISCRIMINANT (10 dim. feature space 5 regions parameter chosen 2->6)	
CODEWORD	TRAIN(%)	TEST(%)
32	93.31	54.40
64	97.18	59.90
128	99.85	58.89
256	100.0	57.91

Table 3-8. Effect of feature selection on the recognition rate. (50 words)
(Transform matrix is based on training speakers
20 speakers for training and
10 speakers for testing.)

3.6 Summary

In this chapter, we discussed detailed information for preprocessing techniques to use with isolated-word HMM ASR systems. We first described the algorithms for these preprocessing techniques and then presented test ASR results for these techniques with an HMM/VQ recognizer and a CVC isolated-word data base. The recognizer was tested under a variety of conditions. Different ways of segmenting the speech waveform according to acoustic phonetic labels and other aspects of the implementation were varied in an attempt to maximize the usefulness of the preprocessing.

For the discrete cosine transform preprocessing, the training results are almost the same as those without preprocessing and the testing results are lower than those without preprocessing. The big difference between training and testing implies insufficient training data was available. The DCT processing is a good method for data reduction, but this data reduction appeared to degrade recognition accuracy.

For the linear discriminant analysis preprocessing, we can improve ASR results when we segment the waveform in the right ways and use the proper number of VQ codewords. If too many phonetic clusters are defined for the discriminant space, the phonetic discriminating power is reduced. Too few phonetic clusters do not represent the speech waveform correctly. A 10% improvement in recognition rate has been achieved in our experiments by basing the LDA transform on long enough speech segment to account for coarticulation effects.

CHAPTER 4

CONCLUSIONS

The first section of this chapter is a summary of the most significant results of this work. In the next section, comparisons between our results and the results of other systems are presented. The final section of this chapter contains suggestions for future work.

4.1 Summary of Experimental Results

1. A five-state, left-to-right hidden Markov model without skip states is the simplest and the most efficient model for the isolated-word data base used in our experiments.

In chapter two, we have shown that the techniques of vector quantization of DCT vectors and hidden Markov modeling can be combined in a simple, straight forward manner to implement a speaker-independent, isolated-word recognizer. A constrained HMM such as a left-to-right model consistently performs better than an unconstrained model. The required number of states in each word HMM needs to be on the order of five. More states do not lead to significant improvements in performance and models with more states require much longer training times. A HMM with jump transitions which skip one state yields slightly lower recognition rates than models

without jump transitions. Therefore we chose five-state, left-to-right HMM's without transitions that skip states in all our experiments.

2. The three-term cosine basis vector expansion technique appears to be a good approach for time-normalization and data reduction applications. However, it does not enhance phonetic differences for HMM word recognition systems.

A three-term cosine expansion not only time-normalizes phonetic segments of different lengths but also encodes dynamic information of the speech signal. However, after the three-term cosine expansion, some useful information is apparently lost. This approach treats all the phonemes the same and does not enhance phonetic contrasts. The transform length and amount of overlap have little effect on HMM recognition rates. We found no benefit of using this type of transformation for HMM preprocessing.

3. Discriminant analysis is an effective way to enhance phonetic differences and to improve phonetic modeling without increasing the amount of training data.

In chapter three, we gave results for LDA transformations as preprocessors for HMM ASR systems. Recognition accuracy is improved by more than 10% for some similar sounding words in comparison with a normal HMM algorithm, even with a small training set. After the discriminant analysis, all phonologically-equivalent sounds are transformed to a tightly clustered region in discriminant space whereas phonologically

different sounds are widely separately. Since acoustic segments, even for the same phoneme, contain different information, we treated acoustic segments as separate classes. Each acoustic segment was then mapped to a different region on the discriminant space. Since discriminant analysis is a statistical technique based on the properties of the data, it tends automatically to scale and to combine the original variables in terms of their importance, while simultaneously taking into account correlations among the original feature components. This technique also automatically sorts the transformed variables according to their discriminating power in the final discriminant space, and therefore, eliminates the need for lengthy combination tests of variables.

4.2 Comparison of Results with Previous Studies

It is usually very difficult to compare the performance of different front-ends since front-ends are developed based on different philosophies and different requirements. Front-ends are also evaluated under different conditions and different data bases. We will compare our results with those from two other existing front-ends. Both of these front-ends were tested with continuous speech recognition systems.

Researchers from BBN System and Technologies Corp (Yu et. al., 1990) used LDA in an attempt to improve recognition accuracy for a discrete HMM system. All experiments were performed on the DARPA resource Management Corpus using the BBN BYBLOS system. The BBN BYBLOS Continuous Speech

Recognition System uses context-dependent phonetic discrete HMM's based on three codebooks. The first codebook contains 14 mel-frequency warped cepstral coefficients (c1-c14) computed every 10 ms directly from the speech power spectrum. The second codebook contains the 14 "differences" of these parameters, derived by computing the slope of a least square linear fit to a five-frame window centered on each frame. Finally, they used a third codebook that has the amplitude-normalized log rms energy and the "difference" of this energy. They divided the 30 features among three codebooks to avoid the training problem associated with high dimensionality. Each codebook is designed using a nonuniform binary clustering algorithm, followed by several iterations of the k-means algorithm. They chose 50 basic phonemes to be discriminated.

They conducted four experiments with variations in the number of codebooks and assignment of linear discriminants to codebooks. In all four tests they concatenated the 14 cepstral coefficients, the 14 "difference" coefficients, and the two normalized energy coefficients, and used LDA to extract a new set of 30 discriminant features. In the first test (30f), they clustered all 30 discriminant features into one codebook which was used in HMM recognition. In the second test (15f,15f), they split the 30 discriminant features into two 15-parameter codebooks. In the third test (15f), they used only the first 15 discriminant features in a single codebook. Finally, in the fourth test (km,15f), they used the standard three codebooks together with a fourth codebook containing the first

15 discriminant features. For all cases tested, LDA did not improve results significantly over the baseline 3-codebook condition.

On the other hand, however, researchers from Texas Instruments Inc. (Doddington, 1989) claimed that a phonetically-sensitive transformation of speech features does yield significant improvements in speech recognition performance. They used a linear transformation of speech feature vectors to discriminate against out-of-class confusion data as a function of phonetic state. They created a metric which enhances discrimination between the true phonetic state underlying the speech data and all other phonetic states which are confusable with the true state. They then evaluated the technique on the TI/NBS connected digit data base. Error rates of 0.5% (1.5%) for unknown-length strings and 0.2% (0.6%) for known-length strings. These error rates are two to three times lower than the original ones.

The system developed and evaluated in this paper is actually quite simple, with a single model per vocabulary word and a linear discriminant transform front-end. Yet it compares favorably with various more complex systems. Our system improves the recognition by more than 17% (reduces the error rate by more than 45%). It performed better than the system tested by BNN, but we did not achieve as good results as in the TI study. Since the standard DARPA data base is available at this moment, it should be used for all future experiments so that the results of different research studies

can be compared objectively. Also, researchers can concentrate more on algorithm development rather than developing their own data base.

4.3 Suggestions for Future Researches

1. Neural Network for Speech Recognition

Conventional classifiers are very sensitive to time alignment and not capable of dealing with coarticulation effects, thus resulting in low recognition rates. HMM classifiers can compensate for time variations but have poor acoustic/phonetic performance. Artificial neural nets can achieve good performance via dense interconnection of simple computational elements. Single-layer nets can implement algorithms required by Gaussian maximum-likelihood classifiers and optimum minimum-error classifiers for binary patterns corrupted by noise. More generally, the decision regions required by any classification algorithm can be generated in a straight-forward manner by three layer feed-forward nets. Computation elements or nodes are connected via weights that are typically adapted during use to improve performance. Moreover, contextual information can be utilized if the net has internal memory. Therefore, artificial neural networks containing short term memory trained as phonetic classifiers offer the promise of preprocessing similar to that possible with LDA, but presumably more powerful. Therefore we would expect improved performance with neural networks as preprocessors for HMM's.

2. Speaker Independent Phone Recognition

Speech recognition based on phonetic units which are smaller than a word is suitable for large vocabulary recognition, because any word model can be generated from the unit. Recently, HMM's have been successfully used for large vocabulary speech recognition based on such phonetic units (Lee and Hon, 1989). Good phonetic decoding leads to good word decoding, and the ability to recognize the English phones accurately will undoubtedly provide the basis for an accurate word recognizer. Moreover, some researchers have found difficulty in integrating higher level knowledge sources with the phonetic decoder. This will hopefully be overcome by more accurate phonetic decoding.

Our future work will focus on the creation of a large-vocabulary speaker-independent continuous speech recognition system based on the methods used in this study.

BIBLIOGRAPHY

Bahl, L. R. and Brown, P. F. (1988). "Speech Recognition with Continuous-parameter Hidden Markov Models," ICASSP-88, 40-43.

Baum, L. E. (1972). "An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Function of a Markov Process," Inequalities, 3, 1-8.

Brown, P. (1987) "The Acoustic-Modeling Problem in Automatic Speech Recognition," PhD Thesis, CMU.

Buzo, A. and Gray, A. H. (1980). "Speech Coding Based Upon Vector Quantization," IEEE trans. ASSP-28, 562-574.

Deng, L. and Lenning, M. (1988). "Modeling Acoustic-phonetic Detail in an HMM-based Large Vocabulary Speech Recognizer," ICASSP-88, 509-512.

Doddington, G. R. (1989). "Phonetically Sensitive Discriminants for Improved Speech Recognition," ICASSP-89, 556-559.

Hanson, B. and Applebaum, T. (1989). "Robust Speaker-independent Word Recognition Using Static, Dynamic and Acceleration Features," (Speech Technology Laboratory, Santa Barbara, California).

Fisher, R. (1923). "Contributions to Mathematical Statistics," Wiley, New York.

Huang, W., Lippmann, R., and Gold, B. (1988). "A Neural Network Approach to Speech Recognition," ICASSP-88, 99-100.

Hunt, M. J. and Lefebvre, C. (1988). "Speaker Dependent and Independent Speech Recognition Experiments with an Auditory Model," ICASSP-88, 215-218.

Hunt, M. J. and Richardson, S. M. (1990). "Use of Linear Discriminant Analysis in a Speech Recognizer," (Marconi Speech and Information Systems, U.K.).

Juang, B. H. and Wong, D. Y. (1982). "Distortion Performance of Vector Quantization for LPC Voice Coding," IEEE trans. ASSP-30, 294-303.

Lee, K. F. and Hon, H. W. (1988). "Large-vocabulary Speaker-Independent Continuous Speech Recognition Using HMM," ICASSP-88, 123-126.

Lee, K. F and Hon, H. W. (1989). "Speaker-independent Phone Recognition Using Hidden Markov Models," IEEE trans. ASSP-37, 1641-1648.

Leung, H. C. and Zue, V. W. (1988). "Some Phonetic Recognition Experiments Using Artificial Neural Nets," ICASSP-88, 422-425.

Levinson, S. E. and Miller, L. G. (1988). "Large Vocabulary Speech Recognition Using a Hidden Markov Model for Acoustic/Phonetic Classification," ICASSP-88, 505-507.

Levinson, S. E., Rabiner, L. R., and Sondhi,M.M. (1983). "An Introduction to the Application of the Theory of Probabilistic Function of a Markov Process to Automatic Speech Recognition," The Bell System Technical Journal, 62, 1035-1074.

Lippmann, R. P. and Martin, E. A. (1988). "Discriminant Clustering Using an HMM Isolated-word Recognizer," ICASSP-88, 48-50.

Martin, E. A., Lippmann, R. P., and Paul, D. B. (1988). "Dynamic Adaptation of Hidden Markov Models for Robust Isolated-word Speech Recognition," ICASSP-88, 52-54.

Martin, E. A. and Paul, D. B. (1987). "Two-stage Discriminant Analysis for Improved Isolated-Word Recognition," ICASSP-87, 709-712.

Murveit, H. and Weintraub, M. (1988). "1000-word Speaker-Independent Continuous-speech Recognition Using Hidden Markov Models," ICASSP-88, 115-118.

Nossair, Z. B. (1989). "Dynamic Spectral Shape Features as Acoustic Correlates for Stop Consonants," PhD Thesis, ODU.

Rabiner, L. R. and Juang, B. H. (1986). "An Introduction to Hidden Markov Models," IEEE trans. ASSP-34, 4-16.

Rabiner, L. R, Levinson, S. E., and Sondhi, M. M. (1983). "On the Application of Vector Quantization and Hidden Markov Models to Speaker-independent, Isolated-word Recognition," The Bell System Technical Journal, 62, 1075-1105.

Rabiner, L. R., Wilpon, J. G., and Soong, F. K. (1989). "High Performance Connected digit Recognition Using Hidden Markov Models," IEEE trans. ASSP-37, 1214-1225.

Shen, C. H. (1988). "Discrete Cosine Transform Analysis for Speaker-independent Automatic Acoustic-phonetic Transcription," PhD Thesis, ODU.

Thomas, W. P. (1983). "Voice and Speech Processing," McGraw-Hill, New York.

Waibel, A. and Hanazawa, T. (1988). "Phoneme Recognition: Neural Networks vs. Hidden Markov Models," ICASSP-88, 107-110.

William, W. C and Paul, R. L (1971). "Multivariate Data Analysis," John Wiley & Sons, Inc.

Yu, G., Russell, W. and Schwartz, R. (1990). "Discriminant Analysis and Supervised Vector Quantization for Continuous Speech Recognition," ICASSP-89, 685-688.

Zahorian, S. A. and Gordy, P. E. (1983). "Finite Impulse Response Filter for Speech Analysis and Synthesis," Proceedings of IEEE International Conf. on ASSP, April 1983.

APPENDIX

A.

ALL 99 WORDS
(missing speakers have been specified)

1.	COB	(c03)(R1O)	2.	BAH	(f08,c01)(O1U)
3.	GOT	(f10)(Q1T)	4.	BEET	(m10)(O2T)
5.	TEAK	(T2R)	6.	PEEB	(S2O)
7.	TUBE	(c10)(T3O)	8.	BOOT	(O3T)
9.	DUPE	(c02,c10)(P3S)	10.	GAP	(Q4S)
11.	CAP	(c04) (R4S)	12.	PAT	(S4T)
13.	BAT	(O4T)	14.	GERP	(Q5S)
15.	BIRD	(O5P)	16.	PERK	(S5R)
17.	BID	(O6P)	18.	PIG	(S6Q)
19.	TICK	(T6R)	20.	DIP	(P6S)
21.	GIVE	(Q6Z)	22.	DEBT	(P7T)
23.	TED	(T7P)	24.	BET	(c09)(O7T)
25.	KEG	(c10)(R7Q)	26.	TALK	(m03,c08)(T8R)
27.	DAUB	(P8O)	28.	DUG	(P9Q)
29.	TUCK	(T9R)	30.	GUT	(Q9T)
31.	BIB	(O6O)	32.	POT	(S1T)
33.	DOT	(f02)(P1T)	34.	TOP	(c01)(T1S)
35.	POD	(c02)(S1P)	36.	POCK	(S1R)
37.	TOG	(f02)(T1Q)	38.	PEEP	(c01)(S2S)
39.	DEEP	(c02)(P2S)	40.	KEEP	(R2S)
41.	GEESE	(c02)(Q2X)	42.	KEYED	(R2P)
43.	PEP	(S7S)	44.	POOP	(S3S)
45.	TOOT	(T3T)	46.	COOP	(R3S)
47.	GOOK	(f02)(Q3R)	48.	PECK	(c10)(S7R)
49.	GET	(c09)(Q7T)	50.	TACK	(T4R)
51.	DAD	(P4P)	52.	TAB	(T4O)
53.	TAG	(T4Q)	54.	BAUD	(f08)(O8P)
55.	DIRT	(c10,c04)(P5T)	56.	CURB	(c09)(R5O)
57.	TURK	(c10,c03)(T5R)	58.	DURG	(P5Q)
59.	PAUP	(c03)(S8S)	60.	KIT	(m03,c04)(R6T)
61.	HOD	(c01)(U1P)	62.	LEAGUE	(m04,c02)(K2Q)
63.	HEED	(f01,m04,c10)(U2P)	64.	SUED	(X3P)
65.	MOOG	(c02)(L3Q)	66.	WHO'D	(c10,f09)(USP)
67.	HAD	(U4P)	68.	HEARD	(U5P)
69.	HID	(c04)(U6P)	70.	WEB	(I7O)
71.	HEAD	(U7P)	72.	BOUGHT	(O8T)
73.	CAUGHT	(R8T)	74.	GAWK	(m03,f08)(Q8R)
75.	CAWG	(f08)(R8Q)	76.	GAWP	(f08)(Q8S)
77.	HAWD	(f08)(U8P)	78.	BUT	(O9T)
79.	PUTT	(S9T)	80.	CUP	(c10)(R9S)
81.	CUB	(R9O)	82.	BUD	(m06)(O9P)

83. HUD (U9P)
84. BOOK (OAR)
85. TOOK (TAR)
86. PUT (SAT)
87. COULD (RAP)
88. GOOD (QAP)
89. HOOD (UAP)
90. BOAT (OBT)
91. DOPE (PBS)
92. GOAD (QBP)
93. CODE (m06)(RBP)
94. POPE (SBS)
95. TOAD (TBP)
96. COKE (RBR)
97. GOAG (QBQ)
98. COAB (RBO)
99. HOED (UBP)

B.

PHONE LIST
(18 PHONES USED IN OUR EXPERIMENTS)

Phone	Example	Index	Code
1. Vowels (11)			
AA	hot	1	1
IY	bee	2	2
UW	boot	3	3
AE	apple	4	4
ER	bird	5	5
IH	bit	6	6
EH	bet	7	7
AO	bought	8	8
AH	up	9	9
UH	book	10	A
OW	boat	11	B
2. Stops (6)			
BB	bit	24	O
DD	dog	25	P
GG	get	26	Q
KK	kit	27	R
PP	pot	28	S
TT	ten	29	T
3. Fricatives (1)			
HH	hot	30	U

C.

PROGRAM NAME LIST
(All those programs are on one HD diskette)

1. Program for codebook generation.
   main program: codbok.for
   subroutines: (1) farth.for
   (2) kmean.for
   (3) rdpar4.for
   command file: codbok.dat
   output file: codbok.out

2. Program for HMMs train and test.
   main program: hmm.for
   subroutines: (1) init.for
   (2) obseq.for
   (3) trn.for
   (4) tst.for
   (5) rdpar4.for
   command file: hmm.dat
   output file: hmm.out

3. Program for Discriminant Analysis front-end.
   main program: dpanal3.reg (dpanal3.spk for spk normalization)
   subroutines: (1) arr3.for
   (2) rdpar4.for
   (3) sort.reg (sort.spk for spk normalization)
   (4) rdhed.for
   (5) covmn1.for
   (6) covmn5.for
   (7) invs.for
   (8) r.for
   libraries: (1) classify.lib
   (2) matrix.lib
   command file: dpanal1.dat

   Program does the transformation.
   main program: dpanal2.for
   Program does the speaker normalization transformation.
   main program: dpanal5.for

4. Program for 3-term cosine expansion front-end.
   main program: dcstrn.for
   cmmand file: dcstrn.dat

D.

COMMAND FILE FOR FRONT.FOR

```
The following is selection variable between type A and B
1
The following are the # of phonemes that make this category
and their code.
1,'6'
# of phonemes and their codes that can be preceding the above
1,'R'
The following are the following phonemes
1,'T'
The following are acoustic segment phontic labels
1,'SV'
START, IEND, DELTAT, NUMFRM, FRMLEN, FSMETH, FCH
0.0,0.0,10,11,20.0,0,1
the set up for analysis types
1,1,10
set up for DCT COEFFICIENTS
0,3.0,2,2,0.5,150.0,6000.0
NUMBER OF PARTITIONS
8
Name of first partition and number of talkers in that
partition.
'E',6
Talker ID's in partition 1
'F01','M02','C01','F02','M04','C02'
Name of second partition and number of talkers in that
partition.
'F',6
The following are the talker codes in partition 2
'F05','M05','C05','M08','C08','F03'
Name of THIRD partition and number of talkers in that
partition.
'G',2
The following are the talker codes in partition 3:
'M06','C10'
Name of second partition and number of talkers in that
partition.
'F',1
The following are the talker codes in partition 2
'F09'
Name of FOURTH partition and number of talkers in that
partition.
'E',7
Talker ID's in partition 4
```

D.

COMMAND FILE FOR FRONT.FOR

```
The following is selection variable between type A and B
1
The following are the # of phonemes that make this category
and their code.
1,'6'
# of phonemes and their codes that can be preceding the above
1,'R'
The following are the following phonemes
1,'T'
The following are acoustic segment phontic labels
1,'SV'
START, IEND, DELTAT, NUMFRM, FRMLEN, FSMETH, FCH
0.0,0.0,10,11,20.0,0,1
the set up for analysis types
1,1,10
set up for DCT COEFFICIENTS
0,3.0,2,2,0.5,150.0,6000.0
NUMBER OF PARTITIONS
8
Name of first partition and number of talkers in that
partition.
'E',6
Talker ID's in partition 1
'F01','M02','C01','F02','M04','C02'
Name of second partition and number of talkers in that
partition.
'F',6
The following are the talker codes in partition 2
'F05','M05','C05','M08','C08','F03'
Name of THIRD partition and number of talkers in that
partition.
'G',2
The following are the talker codes in partition 3:
'M06','C10'
Name of second partition and number of talkers in that
partition.
'F',1
The following are the talker codes in partition 2
'F09'
Name of FOURTH partition and number of talkers in that
partition.
'E',7
```

```
Talker ID's in partition 4
'F04','M03','C03','F06','M07','C04','F10'
Name of 5TH partition and number of talkers in that partition.
'F',3
Talker ID's in partition 5
'M01','F07','C06'
Name of 6TH partition and number of talkers in that partition.
'E',1
Talker ID's in partition 6
'M09'
Name of 7TH partition and number of talkers in that partition.
'F',4
Talker ID's in partition 7
'F08','C07','M10','C09'
The following are the name of the output parameter files
C:\DATA2\WORD60.OUT
The following is the name of the filter coefficient file
FLTCOF.DAT
```