Copyright © 1983 American Telephone and Telegraph Company
THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 62, No. 4, April 1983
Printed in U.S.A.

On the Application of Vector Quantization and Hidden Markov Models to Speaker-Independent, Isolated Word Recognition

By L. R. RABINER, S. E. LEVINSON, and M. M. SONDHI

(Manuscript received September 10, 1982)

In this paper 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 linear predictive coding analysis front end. This is done in the framework of a standard statistical pattern recognition model. Both the vector quantizer and the hidden Markov models need to be trained for the vocabulary being recognized. Such training results in a distinct hidden Markov model for each word of the vocabulary. Classification consists of computing the probability of generating the test word with each word model and choosing the word model that gives the highest probability. There are several factors, in both the vector quantizer and the hidden Markov modeling, that affect the performance of the overall word recognition system, including the size of the vector quantizer, the structure of the hidden Markov model, the ways of handling insufficient training data, etc. The effects, on recognition accuracy, of many of these factors are discussed in this paper. The entire recognizer (training and testing) has been evaluated on a 10-word digits vocabulary. For training, a set of 100 talkers spoke each of the digits one time. For testing, an independent set of 100 tokens of each of the digits was obtained. The overall recognition accuracy was found to be 96.5 percent for the 100-talker test set. These results are comparable to those obtained in earlier work, using a dynamic time-warping recognition algorithm with multiple templates per digit. It is also shown that the computation and storage requirements of the new recognizer were an order of magnitude less than that required for a conventional pattern recognition system using linear prediction with dynamic time warping.

I. INTRODUCTION

There currently exist two standard approaches to isolated word recognition, namely, feature extraction methods and statistical pattern recognition models. A statistical pattern recognition approach has the property of being a nonparametric approach to recognition and therefore is widely used in most commercial and industrial recognizers.[1–6] The feature-based approach to recognition has been primarily used in the (computationally) less expensive systems, and as a basis for recognition of continuous speech (in conjunction with segmentation and labeling algorithms).[4–9]

In the past few years a new approach to speech processing has been proposed, namely, using probabilistic functions of Markov models. This approach has been applied at the Institute for Defense Analyses for speaker recognition,[10] and at Carnegie Mellon University and IBM to solve problems in continuous speech recognition[11,12] with good success. Based on its success in these related areas of speech processing, a question that arises naturally is how well these probabilistic models would work on problems in isolated word recognition.

It is the prime purpose of this paper to provide an answer to the question posed above. Before discussing the approach we have taken to get at the answer, we must first describe the structure of a word recognition system based on (hidden) Markov models (HMM). As in most recognition systems we assume we have a labeled training set of data from which we build a series of Markov models, one for each vocabulary word. Then when we want to recognize an unknown token, we compute a probability score for each word HMM on that token, and choose as the recognized word the one corresponding to the model with the highest probability score (i.e., the most likely word HMM). Techniques for training and scoring such HMMs are discussed both here and in the companion paper.[3]

In a conventional pattern recognition system the unknown test token is time-aligned in turn to each reference pattern via some form of time-warping procedure, typically, dynamic time warping (DTW). By contrast, no such direct alignment is performed in the HMM system; only an indirect time alignment is obtained based on the probabilistic scoring. Thus it is interesting to study the relationship between probabilistic scoring and DTW as applied to isolated word recognition. As we shall see, there is no simple relationship. We will point out several similarities and differences in the two approaches.

The organization of this paper is as follows. In Section II we briefly review the conventional DTW word recognizer based on LPC modeling, since this will be the focus of comparison throughout the paper. In Section III we review the basic ideas behind the use of HMMs for isolated word recognition. It is the purpose of this section to establish notation and terminology that will define the basic parameters of interest in the HMM system. Section III shows that one inherent feature of the HMM recognizer (as we have implemented it) is that the models need a discrete, finite set of observations (input data) to obtain the best model parameters for each word in the vocabulary. A vector quantizer (VQ) was used to transform the continuous set of linear predictive coefficient (LPC) vectors into a finite observation set. Therefore, in Section IV we describe the key ideas behind vector quantization of LPC sets, and discuss the particular implementation that we used. In Section V we describe the overall structure of the HMM isolated word recognizer. In Section VI we describe a series of experiments used to evaluate the performance of the HMM word recognizer and compare it to the performance of a DTW recognizer on the same vocabulary. The effects, on performance, of several parameter variations in the HMM and VQ are also described in this section. In Section VII we discuss the results of the performance evaluation and comparison experiments. The strengths and weaknesses of the HMM word recognizer are discussed, along with computational and storage comparisons of HMM and DTW word recognizers. We attempt, in this section, to determine the fundamental relationships between the HMM and DTW systems.

II. REVIEW OF CONVENTIONAL DTW WORD RECOGNIZER BASED ON LPC MODELING

Figure 1 shows a block diagram of the LPC-based isolated word recognizer.[2,3] The input speech signal, $s(n)$, recorded over a standard dialed-up telephone line, is bandpass-filtered between 100 and 3200 Hz, and digitized at a 6.67-kHz rate. The first step in the processing is

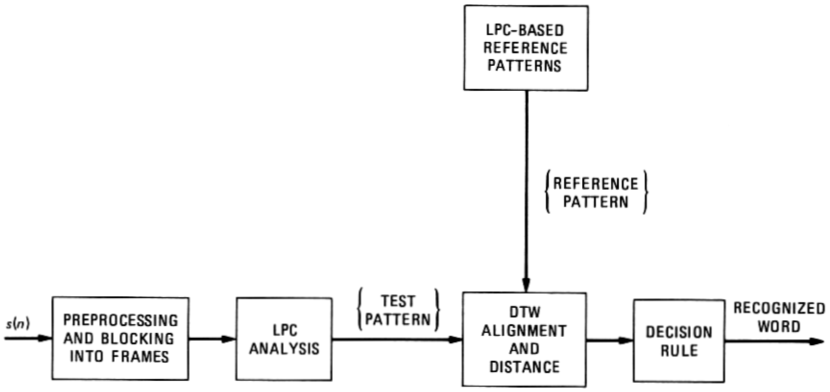

LPC-BASED REFERENCE PATTERNS
REFERENCE PATTERN
s(n)
PREPROCESSING AND BLOCKING INTO FRAMES
LPC ANALYSIS
TEST PATTERN
DTW ALIGNMENT AND DISTANCE
DECISION RULE
RECOGNIZED WORD

Fig. 1—Block diagram of conventional LPC-based word recognizer using a standard dynamic time-warping algorithm for registering test and reference patterns.

the preprocessing block, a first-order digital network, which provides a high-frequency pre-emphasis to the speech. The pre-emphasized signal is blocked into frames of 45 ms (300 samples) with each consecutive frame spaced 15 ms (100 samples) apart. An 8-pole LPC analysis (autocorrelation method) is performed on each frame of the word (after isolating it with an endpoint detector[14]), thus creating the test pattern. This test pattern is compared with each reference pattern using a DTW alignment algorithm that simultaneously provides a distance score associated with the alignment. The distance scores for all the reference patterns are sent to a decision rule, which provides a classification of the spoken word, and possibly an ordered (by distance) set of the best n candidates.

The word reference patterns for the recognizer of Fig. 1 are created by a training algorithm. For speaker-trained applications, typically a single reference pattern is created for each word in the vocabulary using a robust training algorithm.[15] For speaker-independent applications, a set of Q reference patterns is created for each vocabulary word using a clustering procedure.[16,17] Typically, about 12 templates per word are sufficient for recognizing words from a fairly homogeneous adult population of native American talkers.

III. BASICS OF HMM FOR WORD RECOGNITION

We assume we have a finite sequence, $\mathbf{O}$, of observations,

$$\mathbf{O} = O_1 O_2 \cdots O_T, \tag{1}$$

where each observation is a discrete symbol drawn from a finite alphabet of symbols. (For the system we will be describing, the observations are the indices of the LPC vectors obtained from an LPC vector quantizer.) We further assume that the sequence of observations may be modeled as a probabilistic function of an underlying Markov chain whose state transitions are not directly observable; hence the name "Hidden Markov Model." Figure 2 shows such a model, $\mathbf{M}$, which is characterized by the following:

(i) N = the number of states in the model. For the model of Fig. 2, $N = 5$.

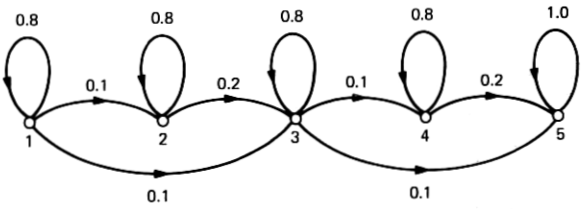


Fig. 2—A typical state diagram for a 5-state Markov model.

(*ii*) M = the number of output symbols in the discrete alphabet of the model. For the present example, $M = 5$.

(*iii*) $A = \{a_{ij}\}$, the transition matrix of the underlying Markov chain. Here, a_{ij} is the probability of making a transition to state j, given that the model is in state i. For the model of Fig. 2 we have

$$A = \begin{bmatrix} 0.8 & 0.1 & 0.1 & 0 & 0 \\ 0 & 0.8 & 0.2 & 0 & 0 \\ 0 & 0 & 0.8 & 0.1 & 0.1 \\ 0 & 0 & 0 & 0.8 & 0.2 \\ 0 & 0 & 0 & 0 & 1.0 \end{bmatrix}.$$

Note that only 11 of the 25 a_{ij}'s are nonzero.

(*iv*) $B = \{B_{jk}\} = \{b_j(k)\}$, the model output symbol probability matrix, where $b_j(k)$ is the probability of outputting symbol k, given that the model is in state j. For the example chosen,

$$B = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0 & 0.5 \\ 0.5 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 & 0.5 \end{bmatrix}.$$

(*v*) $\pi = \{\pi_i\}$, $i = 1, 2, \cdots, N$, the initial state probability vector. For the left-to-right models of the type shown in Fig. 2, we assume the system always begins in state 1, i.e., $\pi_1 = 1$, $\pi_i = 0$, $i \neq 1$.

Isolated word recognition using HMM consists of two phases, training and recognition (or classification). In the training phase, the training set of observations is used to derive a set of reference models of the above type, one for each word in the vocabulary. In the classification phase, the probability of generating the test observation is computed for each reference model. The test is classified as the word whose model gives the highest probability. The computations in each of these phases are fairly straightforward.

Let us begin with the classification phase. Given the observation sequence, **O**, and a model, **M** (i.e., N, M, A, B, and π), the probability of **O** having been generated by model **M** is

$$P(\mathbf{O}|\mathbf{M}) = \sum_{i_1, i_2 \cdots i_T} \pi_{i_1} b_{i_1}(O_1) a_{i_1 i_2} \cdots a_{i_{T-1} i_T} b_{i_T}(O_T). \tag{2}$$

The summation in eq. (2) is more readily computed by defining a forward partial probability, $\alpha_t(i)$, as

$$\alpha_t(i) = P(O_1 O_2 \cdots O_t \text{ and state } i \text{ at time } t|\mathbf{M}). \tag{3}$$

This leads to the recursion

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}), \qquad t = 1, 2, \cdots, T-1 \tag{4}$$

by which eq. (2) can be expressed as

$$P(\mathbf{O}|\mathbf{M}) = P = \sum_{j=1}^{N} \alpha_T(j). \tag{5}$$

In the training phase an initial estimate of the model is made, and P is computed for the training observation sequence according to eq. (5). Next the model is iteratively adjusted to increase P. The iterations are stopped when P stops increasing significantly, or when some other stopping criterion is met (e.g., when the number of iterations exceeds a limit).

One remarkable algorithm for improving a trial model is the Baum-Welch reestimation algorithm.[18] However, maximizing P can also be looked upon as a constrained optimization problem, for which many algorithms have been proposed. In the companion paper in this issue of the Journal,[13] we discuss the relative merits of these procedures.

We now discuss a number of factors that influence the performance of HMM recognizers.

3.1 Initial estimates of A and B

One factor of interest for the HMM recognizer is the choice of initial estimates for the elements of the matrices A and B. The problem here is that although the training procedure is guaranteed to reach a critical point of P, the value of P obtained is typically a local maximum. Hence, alternative starting values of A and B could yield models with higher (or lower) values of P. For our simulations we have chosen to start the training models with essentially random choices for the nonzero elements of both A and B, normalized to satisfy the constraints

$$\sum_{j=1}^{N} a_{ij} = 1 \qquad i = 1, 2, \cdots, N \tag{6a}$$

$$\sum_{k=1}^{M} b_j(k) = 1 \qquad j = 1, 2, \cdots, N. \tag{6b}$$

An alternative starting condition could be

$$a_{ij} = 1/N + \epsilon \tag{7a}$$

$$b_j(k) = 1/M + \epsilon, \tag{7b}$$

where ϵ is a uniformly distributed random variable whose peak is much smaller than either $1/N$ or $1/M$. [Again the a_{ij}'s and $b_j(k)$'s of eq. (7) must be normalized using eq. (6) prior to running the optimization.]

3.2 HMM structures and the number of states

A second factor affecting the determination of optimum HMMs for each vocabulary word is the model structure and the number of states.

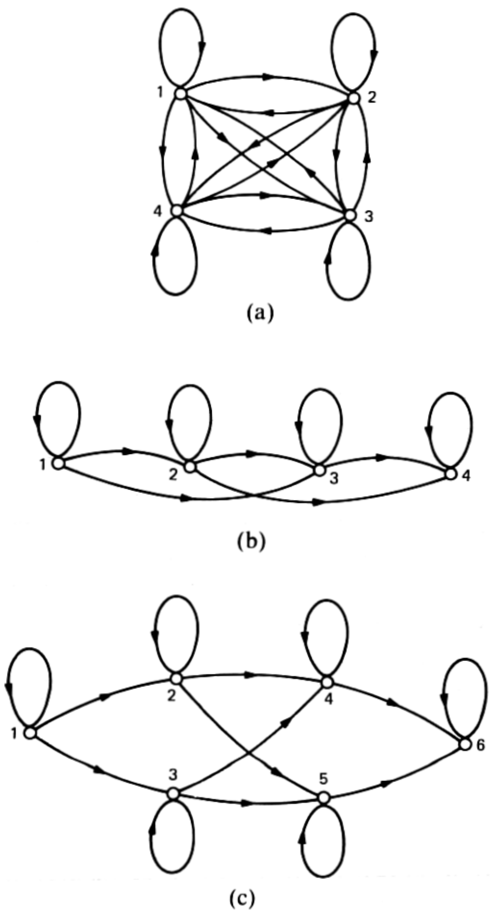


Fig. 3—State diagrams for: (a) unconstrained Markov model with four states, (b) constrained serial Markov model with four states, and (c) constrained parallel Markov model with six states.

We have considered three types of model structures, namely unconstrained, constrained serial, and constrained parallel. Typical examples of each of these models are shown in Fig. 3. In the unconstrained models (shown in Fig. 3a) a transition from any state to any other state can be made—i.e., all a_{ij}'s are allowed to be nonzero. Both the constrained serial models (shown in Fig. 3b) and the constrained parallel models (shown in Fig. 3c) are left-to-right models, i.e., the state transition matrix A is upper triangular. The serial models generally proceed sequentially through the states (although individual states can be skipped over), whereas the parallel models allow multiple paths through the model, with each path skipping one or more model states. For example, there are four distinct paths through the model of

Fig. 3c, 1-2-4-6, 1-2-5-6, 1-3-4-6 and 1-3-5-6, each of which traverses four of the six model states.

Each of the model structures of Fig. 3 can be generalized to include an arbitrary number of states. Recall, however, that the number of free parameters of the Markov model is on the order of N^2 (for the A matrix) plus NM (for the B matrix). Hence, if N gets too large, accurate and reliable determination of the optimum A's and B's may become difficult for a fixed-size training set. However, within these constraints we have investigated models with as few as two states, and as many as 20 states. There appears to be no good theoretical way to choose the number of states needed for a word model, since the states need not be physically related to any single observable phenomenon.

3.3 Multiple observation sequences

A third factor affecting the determination of the optimum HMM for each vocabulary word is the observation sequence used for training. Since we are interested in obtaining speaker-independent models, the observation sequence, $\mathbf{O}$, actually consists of several independent sequences $\mathbf{O}^{(k)}$, $k = 1, 2, \cdots, K$, where $\mathbf{O}^{(k)}$ is the training sequence for talker k, and K is the total number of talkers used for training. Typically, a value of $K = 100$ has been used in our clustering work for speaker-independent training. The way in which we handle multiple sequences is to calculate $P(\mathbf{O}^{(k)}|\mathbf{M})$, using eq. (5), for each sequence, and maximize the product of the probabilities, i.e.,

$$P = \prod_{k=1}^{K} P(\mathbf{O}^{(k)}|\mathbf{M}). \tag{8}$$

The implementation of the computation of eq. (8) is straightforward for the Baum-Welch reestimation procedure, as well as for the gradient methods.[13] Thus the fact that the training data consist of multiple sequences causes no problem in estimating the optimum HMM parameters.

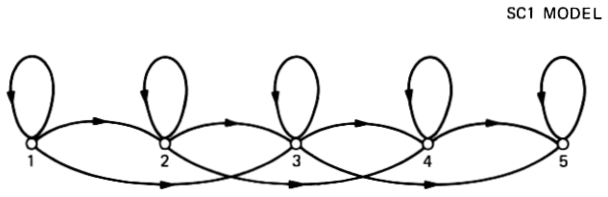
3.4 Constraints on A, B matrices during training

As we show in Fig. 3 we have considered three general HMM structures. For the unconstrained structure the A and B matrix elements are allowed to assume any value consistent with the stochasticity constraints. For the constrained serial models we have considered two general constraints, namely:

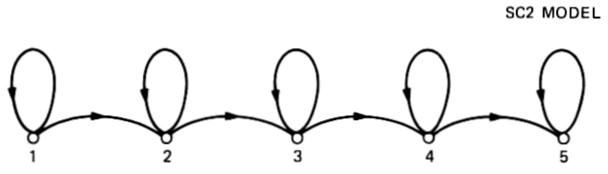
$$\text{SC1: } a_{ij} = 0 \quad \text{for} \quad j < i \quad \text{and} \quad j \geq i + 3 \text{ (double skip allowed)} \tag{9a}$$

$$\text{SC2: } a_{ij} = 0 \quad \text{for} \quad j < i \quad \text{and} \quad j \geq i + 2 \text{ (single skip allowed).} \tag{9b}$$

These two cases are illustrated in Fig. 4 for a 5-state model. Constraints SC1 allow single- or double-state jumps when exiting a given state,



(a)



(b)

Fig. 4—Markov models for two types of serial constraints: (a) single and double transitions allowed, and (b) only single transitions allowed.

whereas constraints SC2 only allow single jumps when exiting a state. Hence, models of the type shown in Fig. 4a have somewhat more flexibility than those of the type shown in Fig. 4b.

For the constrained parallel models we have only considered transition matrices of the type illustrated in Fig. 3c, i.e., a given state can only exit to either of a pair of states in the next column of the grid.

For the most part we have not constrained the B matrix. However, one problem arises if the B matrix is left completely unconstrained. The problem is that a finite training sequence of length T may result in $b_j(k) = 0$. In classification it can then be the case that $\alpha_{t-1}(i)\,a_{ij}$ is nonzero for only one value of j, and $O_t = k$, then the probability of that sequence arising from the model with $b_j(k) = 0$ is $P = 0$; hence a recognition error must occur. This is the so-called missing or inadequate training data problem. We handle this problem (see Ref. 13 for a justification) by using post-estimation constraints on the $b_j(k)$'s of the form

$$b_j(k) \geq \epsilon, \tag{10}$$

where ϵ is a suitably chosen threshold value. All $b_j(k)$'s are compared to the ϵ threshold and those that are below ϵ are replaced by ϵ_j for each j. After this replacement, each $b_j(k)$ that was not changed to the ϵ value is rescaled by the quantity $1 - R_j\epsilon$ [where R_j is the number of $b_j(k)$'s changed for a given j] to properly normalize the $b_j(k)$'s.

It should be clear that all the constrained HMMs are left-right models in that the observations must begin in state 1, must proceed from state to state in a monotonically increasing manner, and must end in state N. Thus, temporal information in the observation sequence is coded directly into the left-to-right HMM.

3.5 *Multiple estimates of A, B, and averaging*

As we mentioned earlier, the reestimation and gradient optimization procedures are guaranteed to find a critical point of P for each HMM. However, in practice, a large number of such points exist in the parameter space. Thus, different initial conditions on A and B may lead to different solutions. To understand the variability in model parameters as well as its effects on overall recognition performance, a series of HMMs were obtained for each word by selecting R random starting sets for A and B, and solving for the optimum A and B in each case. By scoring each of the R models individually, we can obtain an indication of the statistical variability in performance score owing to uncertainty in A and B.

An alternative procedure to using multiple HMMs for each word, obtained from different random starting values for A and B, is to average the R sets of A and B to give an averaged model for each word. The effects of such averaging on word recognition accuracy will be discussed in Section 6.3.

3.6 *Scoring of observation sequences*

One way to score a given observation sequence, $\mathbf{O}$, is to use the iterative calculation of eqs. (4) and (5). We call this the Baum-Welch score, P_{BW}. For left-to-right models, eq. (5) is modified as

$$P_{BW} = \alpha_T(N) \tag{11}$$

because the sequence is constrained to end in state N.

An alternative scoring procedure for the observation sequence, $\mathbf{O}$, given the model, $\mathbf{M}$, is the Viterbi algorithm,[20] which may be compactly stated as:

(*i*) Initialization—$\delta_1(i) = \log[\pi_i b_i(O_1)]$, $i = 1, 2, \cdots, N$

(*ii*) Recursion—for $2 \le t \le T$, $1 \le j \le N$

$$\delta_t(j) = \max_{1 \le i \le N} \{\delta_{t-1}(i) + \log[a_{ij} b_j(O_t)]\}$$

(*iii*) Termination—$P_{VI} = \delta_T(N)$ for left-to-right models

$$= \sum_{j=1}^{N} \delta_T(j) \text{ for unconstrained models.}$$

The above algorithm is a form of the well-known dynamic programming method and can be shown to have the property of determining the state sequence $\mathbf{i} = i_1 i_2 \cdots i_T$, which maximizes

$$P(\mathbf{i} | \mathbf{O}, \mathbf{M}).$$

It is easily shown that both the Baum-Welch and Viterbi scoring procedures require roughly the same amount of computation. The major differences are in the interpretation of the resulting solutions.

IV. VECTOR QUANTIZATION OF LPC COEFFICIENTS

In Section III we noted that in our implementation of HMMs for isolated word recognition the inputs to the model are assumed to be sequences of discrete symbols chosen from a finite alphabet. We obtained these discrete symbols by using the method of vector quantization[19,21] of the LPC vectors measured as described in Section II. In this section we review the theory of vector quantization and discuss its implementation for isolated word recognition.

4.1 Theory of vector quantization

Assume we have a training set of LPC vectors, $\mathbf{a}_i$, $i = 1, 2, \cdots, I$, which are a good representation of the types of LPC vectors that occur when the words in the vocabulary are pronounced by a wide range of talkers. The main idea behind vector quantization is to determine the optimum set of codebook LPC vectors, $\hat{\mathbf{a}}_m$, $m = 1, 2, \cdots, M$, such that for a given M, the average distortion in replacing each of the training set vectors, $\mathbf{a}_i$, by the closest codebook entry, $\hat{\mathbf{a}}_m$, is minimum.

More formally stated, if we define $d(\mathbf{a}_R, \mathbf{a}_T)$ as the distance between two LPC vectors, $\mathbf{a}_R$ and $\mathbf{a}_T$, then the goal of vector quantization is to find the set, $\hat{\mathbf{a}}_m$, such that

$$\|D_M\| = \min_{\hat{\mathbf{a}}_m} \left\{ \frac{1}{I} \sum_{i=1}^{I} \min_{1 \le m \le M} [d(\hat{\mathbf{a}}_m, \mathbf{a}_i)] \right\} \tag{12}$$

is satisfied. The quantity $\|D_M\|$ is the average distortion (distance) of the vector quantizer.

The way in which eq. (12) is solved, for a given value of M, is due to Juang et al.[21] The algorithm first finds the optimum solution for $M = 2$ (two codebook entries), then splits each optimum LPC vector into two components, and finds the optimum solution for $\hat{M} = 2 \cdot M$. This procedure iterates until M is as large as desired. A flow diagram of the details of the codebook generation procedure is given in Ref. 21. The local distance used in our system was the likelihood distance,[2]

$$d(\mathbf{a}_R, \mathbf{a}_T) = \frac{\mathbf{a}_R V_T \mathbf{a}_R'}{\mathbf{a}_T V_T \mathbf{a}_T'} - 1, \tag{13}$$

where V_T is the autocorrelation matrix of the sequence that gave rise to LPC vector $\mathbf{a}_T$.

4.2 Implementation of the vector quantizer

To train the vector quantizer we used a set of 39708 LPC vectors, obtained by using all the vectors in one complete set of 10 isolated digits uttered by each of 100 talkers (50 male, 50 female). Applying the algorithm of Juang et al.,[21] we generated vector quantizers of size $M = 2, 4, 8, 16, 32, 64$, and 128. During the course of running the algorithm, several performance criteria were monitored, including:

(*i*) Average distortion, $\|D_M\|$, of eq. (12)

(*ii*) Sigma ratio (cluster separation) of the resulting codebook entries (clusters), defined as

$$\sigma = \frac{\frac{1}{M}\sum_{i=1}^{M}\left(\frac{1}{M-1}\right)\sum_{j=1}^{M} d(\hat{\mathbf{a}}_i, \hat{\mathbf{a}}_j)}{\|D_M\|}, \tag{14}$$

where the numerator is the average intercluster distance, and $\|D_M\|$ is the average intracluster distance.

(*iii*) Cluster cardinality, N_i, defined as the number of tokens in the ith cluster (i.e., the cluster represented by the ith codebook entry).

(*iv*) Cluster distortion, $\tilde{d}_i$, defined as the average distortion (distance) for the ith cluster.

It should be clear that the average distortion, $\|D_M\|$, satisfies the relation

$$\|D_M\| = \frac{1}{I}\sum_{i=1}^{M} \tilde{d}_i \cdot N_i \tag{15}$$

and that the cluster occupancy satisfies the relation

$$\sum_{i=1}^{M} N_i = I. \tag{16}$$

Results of running the VQ algorithm on the training set of 39708 vectors are given in Figs. 5 through 8. Figure 5 shows plots of $\|D_M\|$ versus M (on a log scale) (part a), and the σ-ratio versus M (part b) for values of M from 2 to 128. We can see that for values of $M \geq 32$ the average distortion falls below 0.3, and that for $M = 64$ the value of $\|D_M\| \approx 0.2$. If we use the conventional recognizer of Fig. 1, the average distance between repetitions of a word (after DTW alignment) has been found to be on the order of 0.3 to 0.4;[16] hence, values of $\|D_M\| <$ 0.3 imply smaller error for the VQ than for interreplication variations of words. The σ-ratio plot shows ratios greater than 10 for $M \geq 32$; hence, extremely good cluster separation is achieved in the vector quantizer for these values of M.

Figures 6 through 8 show a detailed analysis of the statistics of the vector quantizer output for $M = 128$. Figure 6a shows the cluster

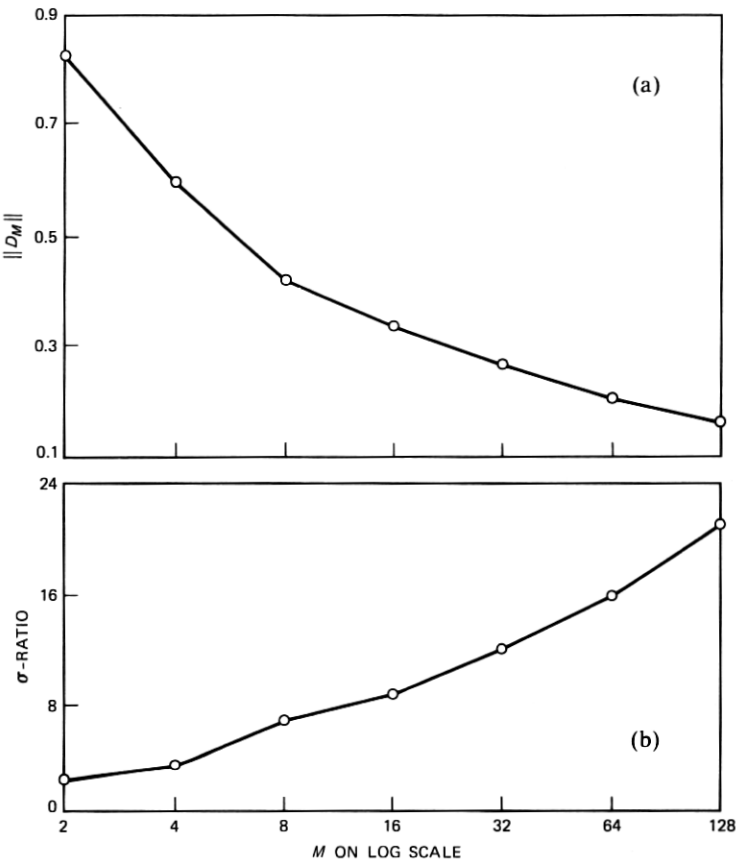


Fig. 5—Plots of vector quantizer performance versus size of codebook for: (a) average distortion, and (b) sigma ratio.

cardinality as a function of the VQ index, and Fig. 6b a histogram of cluster cardinality. The largest cluster has 857 tokens, whereas the smallest cluster has 119 tokens; hence, a spread of over 7 to 1 in cluster occupancy is obtained. The average cluster cardinality, for this case, is 310 tokens, as denoted by the dashed line in Fig. 6a. The histogram of cluster cardinality indicates that the vast majority of clusters have fewer than the average number of tokens.

Figure 7a shows the cluster distortion as a function of the VQ index, and Fig. 7b shows a histogram of cluster distortions. The largest distortion for any cluster is 0.303, whereas the smallest distortion is 0.047; hence, a spread of more than 6 to 1 is observed in cluster distortions. The dashed lines in Fig. 7 denote the average cluster distortion, which in this case is 0.165.

Finally, Fig. 8 shows a plot of the total cluster distortion, defined as

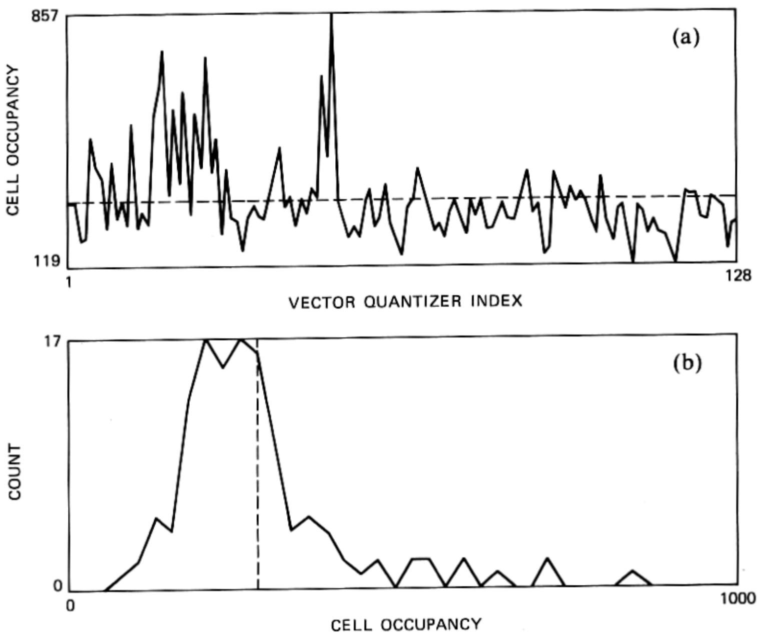


Fig. 6—Plots of (a) cell occupancy versus codebook index, and (b) histogram of cell occupancy for 128-codeword vector quantizer.

$N_i\tilde{d}_i$, versus VQ index. The range of total cluster distortion is from 25 to 72; hence, a spread of less than 3 to 1 is obtained. It is conjectured that the "ideal" vector quantizer seeks to determine the set of "optimum" codebook vectors such that the total cluster distortion is as close to uniform as possible. Hence, clusters with large average distortions should have low cardinality, whereas clusters with small average distortion should have high cardinality. It can be seen from Figs. 6 through 8 that the total cluster distortion statistics are much closer to uniform than are either the cardinality or the average cluster distortion statistics.

Based on the results shown in Figs. 5 through 8, it was decided to implement the HMM recognizer using a $M = 64$ VQ, since the small decrease in average distortion from $M = 64$ to $M = 128$ did not justify the increased computation owing to the larger codebook.

Figure 9 shows some properties of the LPC vectors in the codebook for $M = 64$. Shown in this figure are plots of the first few resonances of the 64 codebook entries (part a), and plots of first versus second resonance (part b), first versus third resonance (part c), and second versus third resonance (part d). As we anticipated, typical vowel resonances for the digits are seen clearly in the plots (e.g., high front

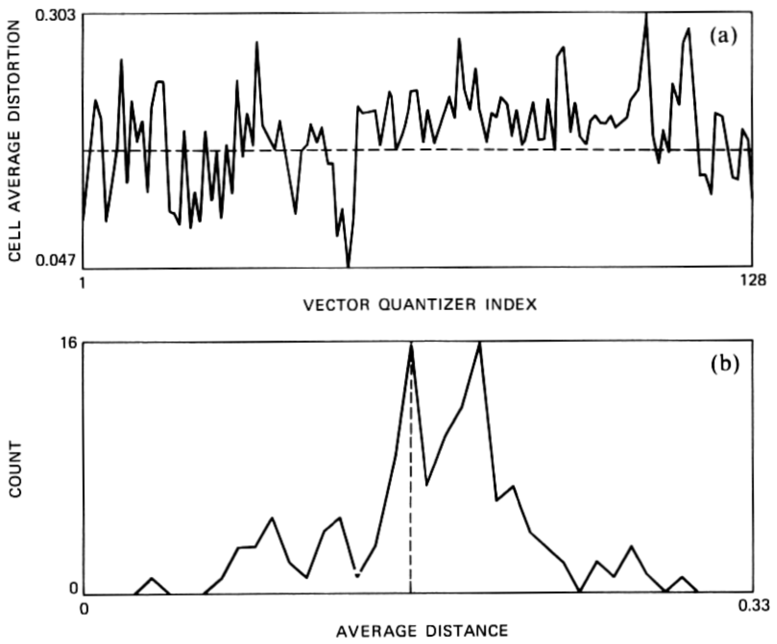


Fig. 7—Plots of (a) cell average distortion versus codebook index, and (b) histogram of cell average distortion for 128-codeword vector quantizer.

vowels, low back vowels, etc), along with characteristic resonances of transient and other nonvoiced sounds. Detailed examination of the spectra of the 64 codebook entries did not provide any further enlightenment as to the VQ properties.

V. OVERALL HMM/VQ ISOLATED WORD RECOGNIZER

A block diagram of the overall HMM/VQ isolated word recognizer is given in Fig. 10. The recognizer operates as a speaker-independent word recognizer, which runs first in a training mode, to provide the codebook entries of the VQ, and the model coefficients of each word HMM.

In the classification mode the LPC sets of the unknown word are first sent through the vector quantizer (to give a finite set of VQ indices) and then scored on each word HMM (using either the Viterbi scoring or the Baum-Welch scoring) to give a probability score for each word model. The decision rule chooses the word whose model gives the highest probability.

In the next section we describe the results of several tests designed to measure the performance of the HMM/VQ word recognizer and to compare it with that of a conventional LPC/DTW recognizer.

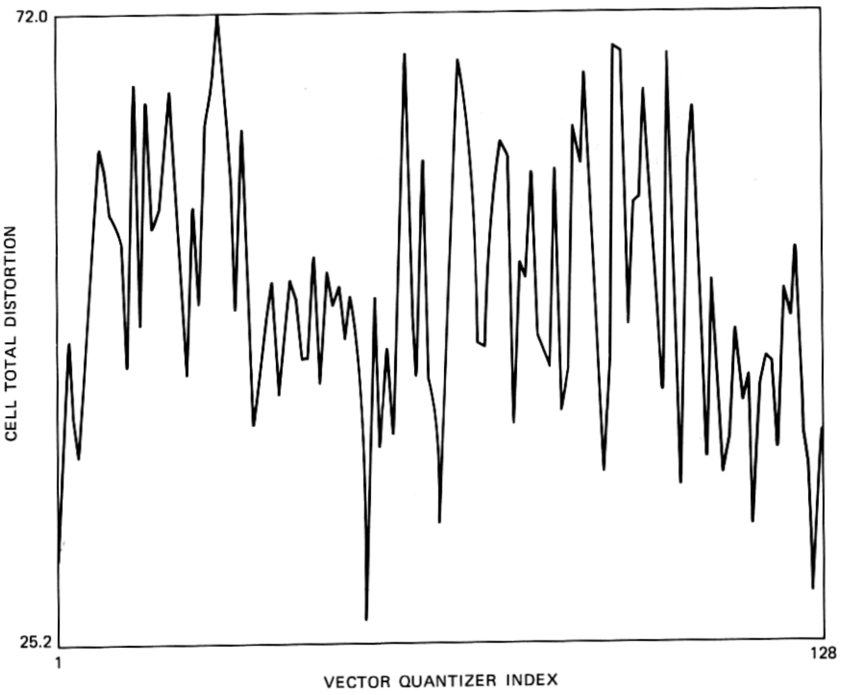


Fig. 8—Plot of cell total distortion versus codebook index for 128-codeword vect quantizer.

VI. EVALUATION EXPERIMENTS AND RESULTS

Several evaluation tests were performed on the HMM/VQ and LPC/DTW isolated word recognizers. For most conditions a single test set of data, denoted as TS1, was used, consisting of one replication of each of the 10 digits by a set of 100 talkers. These talkers were the same ones used to train the recognizer; however, the test replication was recorded many days after the training replication. A second test set of data, denoted as TS2, was used in a couple of tests. This test set consisted of 20 replications of each of the 10 digits by a set of 10 new talkers (5 male, 5 female). Thus, TS2 contained twice as many test tokens as TS1, but represented only one-tenth the number of talkers; however, none of the talkers was included in the training set for either the VQ or the HMMs.

The results presented in this section are the output of a series of recognition tests in which one or more features of the HMM/VQ recognizer were varied. Following the presentation of the results of each of the individual experiments, we shall endeavor to provide a measure of coherency to the results.

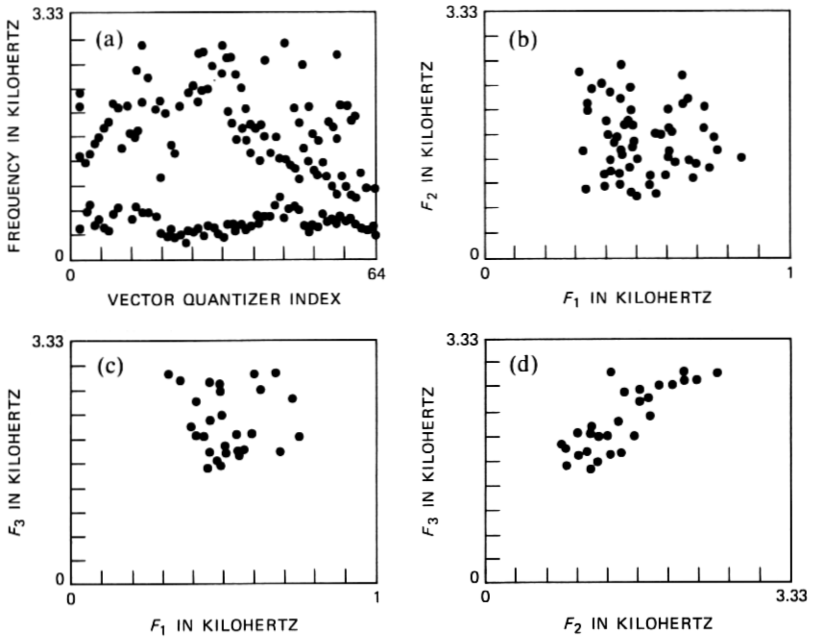


Fig. 9—Plots of locations of first three resonances of codebook vectors: (a) plotted as a function of the vector index, (b) plotted in the $F_1 - F_2$ plane, (c) plotted in the $F_1 - F_3$ plane, and (d) plotted in the $F_2 - F_3$ plane.

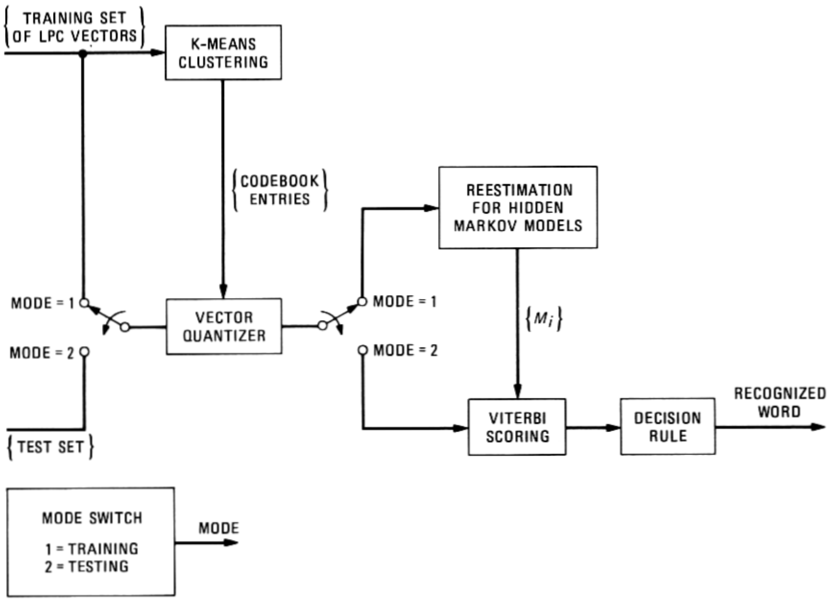


Fig. 10—Overall block diagram of the hidden Markov model—vector-quantizer isolated word recognizer.

6.1 Effects of constrained and unconstrained HMMs

The first set of experiments sought to understand the effects of placing constraints on the A and B matrices on the performance of the overall recognizer. As such, the HMM was trained for an $N = 5$ state model with the following constraints:

UC model:	No constraints placed on A; epsilon constraints [of the type given in eq. (10)] placed on B.
CO model:	The constraints of eq. (9a) placed on A; epsilon constraints of eq. (10) placed on B.
UC.35 model:	Same as UC model but all training sequences with VQ distortions greater than 0.35 were eliminated.
CO.35 model:	Same as CO model but all training sequences with VQ distortions greater than 0.35 were eliminated.

For each of the above four models, the 1000-digit sequences of TS1 were used to measure the overall error rate as a function of the ϵ constant parameter. The results are given in Fig. 11, which shows plots of error rate versus ϵ (on a log scale) for each of the four models. Several trends clearly emerge from these results. First, we can see that a nonzero value of ϵ is an absolute *necessity* for obtaining good performance. Whenever a symbol, k (a VQ index), appears in a test word in a state, j, where $b_j(k) = 0$, the probability for that word model is multiplied by the ϵ value. If $\epsilon = 0$ then the word model is eliminated from consideration and an error occurs. For finite, nonzero values of ϵ, however, such errors need not, and generally will not, occur. Hence,

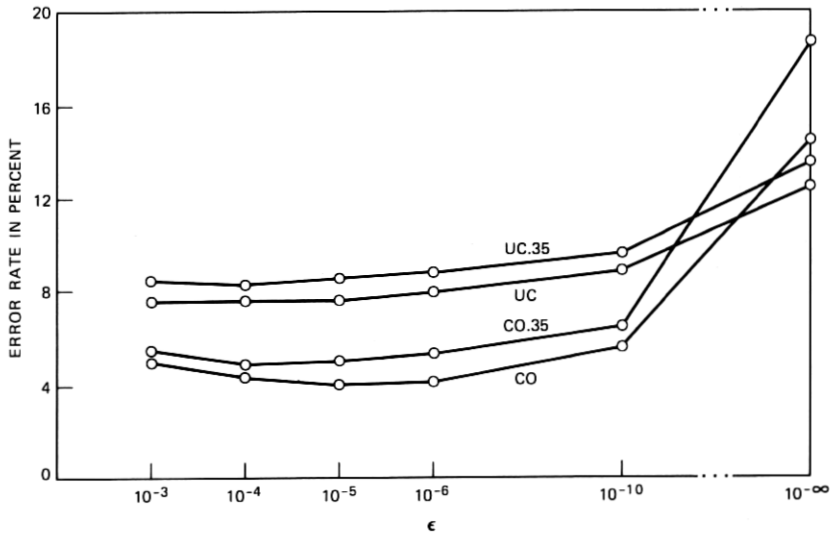


Fig. 11—Plots of average word error rate versus the minimum value of the symbol probability matrix, ϵ, for four types of hidden Markov models, for TS1 data.

even for ϵ on the order of 10^{-10}, the error rate is substantially smaller than for $\epsilon = 0$.

A second clear trend that can be seen in Fig. 11 is that the constrained serial models performed consistently better than the unconstrained models for all nonzero values of ϵ. This result implies that the extra freedom of the unconstrained models tends to raise the probability scores for incorrect words more than for correct words. This is somewhat reminiscent of the fact that opening up the search region of a conventional DTW search helps the wrong words much more than it does the correct words.[16]

It can also be seen from Fig. 11 that it is *always* preferable to train with all available sequences. This result suggests that the more training data given to the HMM model estimation algorithm the better the estimates of the HMM parameters, even if some of the data are less than ideal.

The final trend that emerges from the curves of Fig. 11 is that there is a large range of values of ϵ for which essentially identical performance results. For example, in the range $10^{-10} \leq \epsilon \leq 10^{-3}$, for model CO, the recognition error rate changes by less than 1.6 percent. Thus, so long as ϵ is in this broad range, the exact value of ϵ is not overly significant.

Based on the results of this first series of experiments, we applied the following restrictions:

(*i*) Consider only constrained HMMs

(*ii*) Constrain B matrix entries such that $b_j(k) \geq \epsilon = 10^{-5}$

(*iii*) Use all possible training sequences for the HMMs.

Before we proceed to the next series of experiments, some comments should be made about practical methods of implementing constraints on the B matrix. In Ref. 13 we show how the constraints on the $b_j(k)$ coefficients, of the type given in eq. (10), can be incorporated directly into either the gradient or the Baum-Welch reestimation algorithm. We have also tried a post-normalization technique in which no constraints were placed directly on the B matrix. Following convergence, the B matrix was examined and all entries whose values were below ϵ were reset to the value ϵ, and the rows of the matrix were suitably renormalized to sum to 1.0. Our recognition results indicate identical performance for both the direct and the post-normalization constraint methods. Hence, there appears to be no advantage to constraining the B matrix directly.

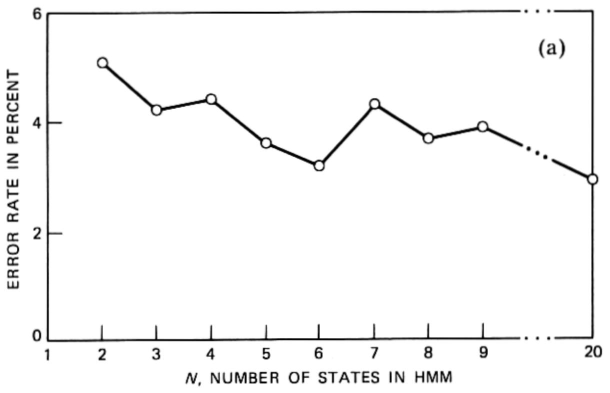
6.2 Markov model with variable number of states

The second set of experiments consider the effects on recognition accuracy of using the constrained serial HMMs with different numbers of states. In particular we computed the optimum SC1 model (see Fig.

4) for each digit where the number of states varied from 2 to 9. We also computed the optimum SC1 model for a 20-state model for each digit.

To evaluate these different models a recognition test was conducted in which each digit was represented by a single N-state HMM, where N took on the values 2 to 9, and 20. The results of this experiment are given in Fig. 12a, which shows word error rate versus number of states in the HMM for the data of TS1. We see a steady but slow decrease in the average word error rate in this curve. We also see a statistical fluctuation in the curve owing to the sampling variability in the A's and B's for each word HMM. (We will return to this issue later in this section.)

A second recognition test was conducted in which each word was represented by *all* the word models with number of states up to some maximum value, *NMAX*. The results of this recognition test (again using TS1 data) are shown in Fig. 12b. A somewhat smoother curve of



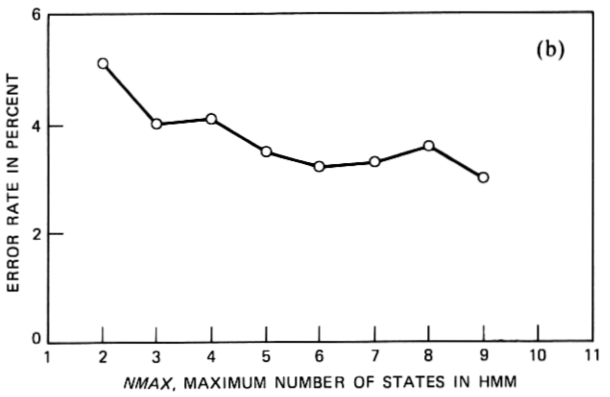


Fig. 12—Plots of (a) error rate versus the number of states in the HMM, and (b) error rate versus the maximum number of states in the HMMs for TS1 data.

errors versus *NMAX* is obtained in Fig. 12b than for the individual models of Fig. 12a; however, the general behavior of both curves is similar.

Figure 13 shows a breakdown of the error rates for each digit for the experiment in which a single HMM, with *N* states, was used for each digit. There is a highly complex interaction between error rate and model size for all digits. Thus it cannot be argued, for instance, that digits like zero and seven (two syllables) need more states in their models than digits like two or one (monosyllables).

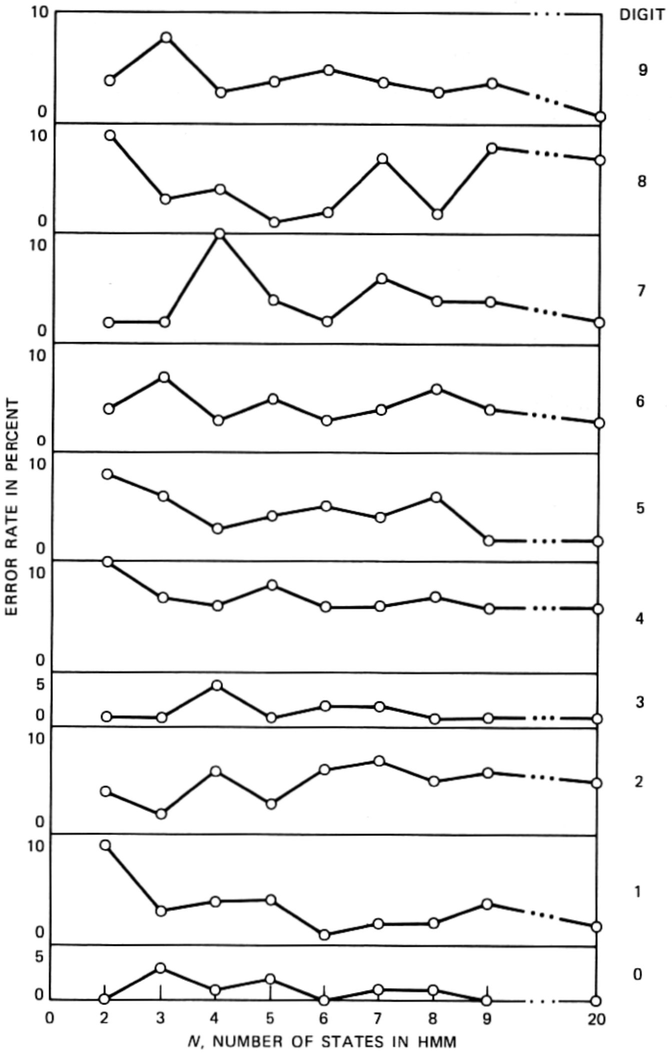


Fig. 13—Individual plots of digit error rates versus the number of states in the HMMs.

From the results shown in Figs. 12 and 13, it was concluded that there is very little gain in using HMMs with more than five or six states when the SC1 structure for each model is being used. It was also concluded that no simple relationship existed between word (digit) accuracy, number of sounds (syllables, etc) in the word, and number of states needed in the word HMM.

6.3 Effects of random starting points

The third set of experiments was concerned with the statistical variability in the performance scores resulting from statistical variability in the parameters of the word HMMs because of different random initial estimates of the parameters. To quantify this effect, a 5-state SC1 model was generated for each digit using 10 different random starting sets of model parameters. Thus, for each digit, 10 "equivalent" HMMs were created.

A recognition test was then run, using TS1 data, in which each of the 10 models was tested separately. Also tested was the case in which all 10 models were used for each digit, as well as the case in which a single model was used for each digit, where the model parameters were obtained by averaging the parameter estimates for each of the 10 word models. The results of these recognition tests are given in Fig. 14, which shows word error rate versus the random start number. Also shown, as single isolated values, are the error scores for the average and the combined 10 state runs. The dashed line in Fig. 14 is the average error rate of the 10 individual models.

From the data of Fig. 14, we can see that the 10 individual models all performed identically to within ±1 percent; hence, the expected statistical variability in error rate scores, due to random starts, should

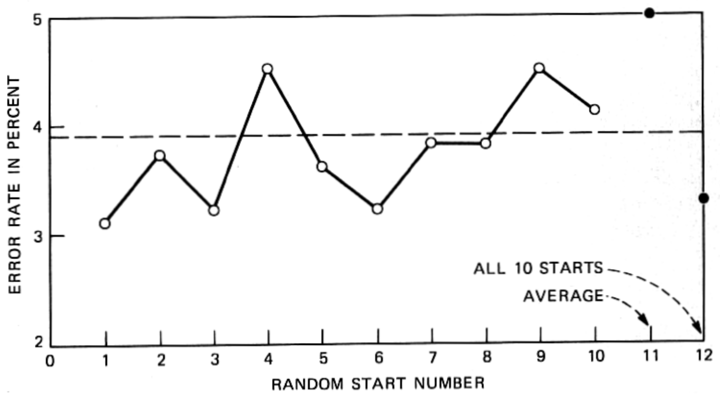


Fig. 14—Plots of error rate for 10 different random starting sets of values of the HMM parameters. Also shown are individual error rates for an averaged model and for combining 10 random start models.

be on the order of ±1 percent. We can also see that the performance of the "averaged" model (obtained by averaging HMM parameters for each word) was somewhat poorer than that of any of the 10 individual models. The fact that using all 10 word models for each digit gives an error rate comparable to that of the *best* word models indicates that multiple word models provide a small gain that comes at the cost of greatly increased computation.

Overall, the data of Fig. 14 suggest that a single model per word should be adequate for most purposes, and that the effects of different random starts on the overall error performance are small.

6.4 Parallel constrained HMMs

The last factor investigated was the effect of using constrained parallel HMMs for each digit. The main idea was that a true parallel structure could model the effects of using a multiplicity of word models in much the same way as multiple templates are used in the conventional LPC/DTW word recognizer.

Figure 15a shows the 5-state constrained serial HMM that was used in previous experiments along with a 7-state, constrained parallel HMM (Fig. 15b), and an 8-state, constrained parallel model (Fig. 15c). The 7-state parallel structure was intended to represent four distinct 5-state word models, in that there were four sets of paths through the model, namely, 1-2-5-7, 1-2-6-7, 1-3-5-7, and 1-3-6-7. The 8-state parallel structure was intended to represent eight distinct 5-state word models in that there were eight sets of paths through this model, namely, 1-2-4-6-8, 1-2-4-7-8, 1-2-5-6-8, 1-2-5-7-8, 1-3-4-6-8, 1-3-4-7-8, 1-3-5-6-8, and 1-3-5-7-8.

Each of the three HMM structures of Fig. 15 was used to generate a word model for each of the digits. The three sets of models were then tested using TS1 data. The results showed each of the three systems obtained the same word error rate (3.5 percent) to within ±0.1 percent. These results indicated that there was really no advantage in using the parallel structure.

6.5 Comparison with LPC/DTW recognizers

To provide some basis of comparison for the performance of the HMM/VQ recognizer with that of more conventional word recognizers, the data of TS1 was tested on the LPC/DTW recognizer of Fig. 1. The reference set consisted of 12 templates per digit, generated from a clustering analysis of the 100 tokens of each digit in the training set (the same training set used to train each word HMM). The decision rule was the nearest neighbor rule (KNN = 1).[17]

Tables I and II show average word recognition accuracies for the following three recognizers:

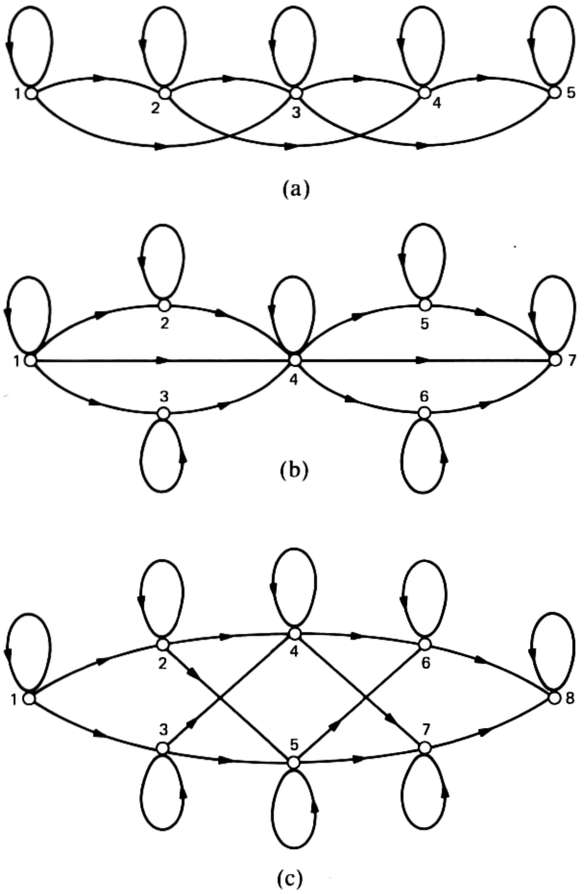


Fig. 15—State diagrams for (a) simple constrained serial 5-state model, (b) constrained parallel 7-state model, and (c) constrained parallel 8-state model.

(*i*) HMM/VQ using a constrained serial structure with five states for the HMM and the 64-element VQ.

(*ii*) LPC/DTW, the conventional recognizer.

(*iii*) LPC/DTW/VQ, the conventional recognizer with both reference and test patterns quantized using the same VQ used in the HMM case.

The results shown in Table I are for the 1000 digits of TS1, whereas those shown in Table II are for the 2000 digits (10 talkers) of TS2. The results given in Table I show that for TS1, both the LPC/DTW and HMM recognizers, when using the VQ, achieved essentially the same digit accuracy; however, the LPC/DTW system without the VQ achieved a 2-percent higher word accuracy. For TS2 in Table II the

Table I—Comparison of results on HMM/VQ and LPC/DTW word recognizers for [average word accuracy (%)] TS1, 100 talkers, 10 digits per talker

	Recognizer		
Digit	HMM/VQ	LPC/DTW	LPC/DTW/VQ
0	98	99	99
1	98	98	99
2	96	100	96
3	99	99	97
4	93	97	96
5	97	96	93
6	96	100	94
7	99	100	94
8	92	98	96
9	95	98	96
Average	96.3	98.5	96.5

Table II—Comparison of results on HMM/VQ and LPC/DTW word recognizers for TS2, 10 talkers, 200 digits per talker

	Recognizer		
Talker	HMM/VQ	LPC/DTW	LPC/DTW/VQ
1	74.5	96	87
2	99.5	100	99.5
3	94	99	97
4	89	99	91
5	95.5	100	100
6	95.5	99	99.5
7	100	100	99.5
8	91.5	96	93
9	91.5	99.5	93.5
10	96.5	98.5	98
Average	92.8	98.7	95.5

results show that the VQ led to a 3.2-percent reduction in accuracy for the LPC/DTW system, and an additional 2.7-percent loss in accuracy for the HMM system. A good deal of the loss in accuracy, however, was contributed by talker 1, whose accuracy was 87 percent for the LPC/DTW/VQ recognizer, and 74.5 percent for the HMM/VQ recognizer. With only 10 talkers, the influence of a single talker on the overall accuracy may be substantial.

An analysis of the actual errors of all three of the recognizers of Tables I and II shows the following:

(*i*) Of the 37 tokens misclassified by recognizer HMM/VQ in TS1, 31 were correctly identified by the LPC/DTW recognizer, and 27 were correctly identified by the LPC/DTW/VQ recognizer.

(*ii*) The vast majority (25) of the 37 errors made by recognizer HMM/VQ were cases in which the probability of the correct word was lower than the probability of the incorrect word by a factor of e^6 or larger.

The results show that when the LPC/DTW recognizer (either with or without the VQ) has incorrectly identified the word, most of the time the HMM recognizer has correctly identified the word. Hence, a side result of this work is that the HMM/VQ model can be combined with an LPC/DTW model to provide word accuracies greater than either individual recognizer could obtain. In particular, for the data of TS1, the combined recognizer could have achieved a 99.4-percent word accuracy by using appropriate decision logic[22] on all cases in which both recognizers did not agree.

The same sort of trends are noted in the errors of TS2. Of the 124 errors made by the HMM/VQ recognizer, 113 were correct in the LPC/DTW or equivalently of the 26 errors made by the LPC/DTW recognizer, 15 were correct in the HMM/VQ system. Hence, again a combined system could potentially achieve an accuracy of about 99.5 percent on TS2 data. Considering that even the talkers in the TS2 data were different from those in the training set, this accuracy appears to be quite remarkable.

In summary, comparisons between the HMM/VQ and LPC/DTW recognizers indicate that without the VQ, the LPC/DTW recognizer achieves from 2- to 6-percent higher accuracy than the HMM/VQ system; with the VQ the differential in accuracy is from 0 to 3 percent.

VII. DISCUSSION

In this paper we showed that the techniques of vector quantization of LPC vectors and hidden Markov modeling can be combined in a simple, straightforward manner to implement a speaker-independent, isolated word recognizer. With adequate training of the vector quantizer and the Markov model estimation algorithm, a digits vocabulary can be recognized, with accuracies of from 93 to 96 percent across a wide variety of talkers. Direct comparisons with a conventional linear predictive coding recognizer using dynamic time warping for time alignment with multiple templates for each vocabulary word showed that the HMM/VQ recognizer performs only a little worse (0.2 percent in one test, 2.7 percent in another) than the LPC/DTW recognizer when using the VQ. Without the VQ, the LPC/DTW recognizer was about 2 to 3 percent better than when the VQ was used.

Several general conclusions can be drawn from the results. The first is that the HMM/VQ recognizer performed exceedingly well on the difficult task of speaker-independent recognition of isolated digits. The fact that the overall performance of the HMM/VQ recognizer was somewhat poorer than the LPC/DTW/VQ recognizer appears to be primarily because of the insufficiency of the HMM training data.

Although 100 training sequences per word are adequate for a clustering analysis, as used in the conventional recognizer, they appear to be inadequate for obtaining good HMM models for these words. This suggestion is made plausible by considering what is being estimated in the HMM for each word. For an N-state Markov model, with M finite symbols per state, a total of $N^2 + NM$ parameters must be estimated. (Of course with constraints there are somewhat fewer parameters.) For $N = 5$, $M = 64$, we need 345 parameters to be estimated from about 100×40 frames of VQ indices. The "curse of dimensionality" would imply that this amount of training data is woefully inadequate. [We have seen one consequence of this inadequacy in that we had to use the ϵ-constraints of eq. (10) on any $b_j(k)$ whose value fell below the ϵ threshold.] In view of this, the fact that we achieve the results we are getting is rather remarkable.

A second conclusion that can be drawn from the results is that the use of the VQ on the LPC sets leads to a small, but not insignificant, degradation in performance of both the HMM/VQ and LPC/DTW/VQ recognizers as compared to the conventional LPC/DTW system. This suggests the need for using more than 64 vectors in the codebook or resorting to continuous models of the LPC parameters.

The results have shown that the errors made by the HMM/VQ and LPC/DTW recognizers are largely disjoint. Here there exists the potential of using some fairly standard techniques to combine the two recognizers into one whose accuracy is as good as the best of both recognizers on any given word.[22] This topic merits further consideration.

The experimentation with various forms of the Markov models used in the recognizer showed fairly conclusively that:

(*i*) Constrained models (with constrained transition matrices) performed consistently better than unconstrained models.

(*ii*) A finite minimum constraint on the state symbol probability matrix was a necessity for good system performance.

(*iii*) The effects of different random starting values for the HMM parameters were negligible in evaluating overall recognizer performance.

(*iv*) The required number of states in each word HMM needed to be on the order of 5. More states did not lead to significant improvements in performance.

(*v*) Parallel HMM structures yielded no real improvements over cascade structures, thereby indicating that an equivalent of multiple HMMs is not readily obtainable by simply changing the model structure.

(*vi*) The Viterbi scoring and the Baum-Welch scoring of test sequences give essentially identical performance.

7.1 Computational considerations of the HMM/VQ recognizer

It is worthwhile estimating the storage required and the computation needed to process an unknown test utterance using the HMM/VQ recognizer, and to compare them to the requirements of the conventional LPC/DTW recognizer. Our intention is to provide only a rough estimate of computational expense. A number of straightforward reductions in computation can be achieved for each recognizer through the judicious use of table storage. Also we ignore overhead owing to index computation, etc.

The first extra step in the HMM/VQ recognizer (after conventional LPC analysis) is vector quantization of the unknown test pattern. If we assume there are T frames in the test word, and M codebook entries in the VQ, then we need a total of

$$C_1 = M \cdot T(p + 1) \tag{17}$$

multiplications* (where p = LPC order) to perform the $M \cdot T$ dot products required to get the best codebook entry for each frame.

Evaluation of the word HMM score, using the Viterbi scoring method with the constrained A matrix, requires approximately

$$C_2 = T \cdot N \cdot 3 \tag{18}$$

multiplications and logarithms per word model, where N is the number of states in the model, and the factor 3 accounts for the number of valid transitions into a given state. For a vocabulary of V words a total of

$$C_* = M \cdot T \cdot (p + 1) + V \cdot T \cdot N \cdot 3 \tag{19a}$$

$$C_{\log} = V \cdot T \cdot N \cdot 3 \tag{19b}$$

multiplications and logarithms are required. For $M = 64$, $T = 40$, $V = 10$, $N = 5$, and $p = 8$, eq. (19) gives $C_* = 29040$ multiplies and $C_{\log} = 6000$ logarithms.

For a conventional LPC/DTW recognizer with Q templates per vocabulary word, the computation for DTW processing is

$$C_3 \cong Q \cdot V \cdot T^2/3 \cdot (p + 1), \tag{20}$$

which for $Q = 12$ and other parameters the same as above gives $C_3 = 576000$ multiplications. Hence, the HMM/VQ recognizer requires about 17 times less computation (assuming logarithms are equivalent to multiplications) than the LPC/DTW recognizer.

With regard to storage, the HMM/VQ recognizer requires

* In this simplified analysis we neglect additions and comparisons of data and use multiplication count as the measure of computation.

$$S_1 = M(p + 1) \tag{20a}$$

$$S_2 = (M \cdot N + 3N) \cdot V, \tag{20b}$$

where S_1 is the storage (in words*) for the VQ codebook entries, and S_2 is the storage (in words) for the set of V word HMMs. For the given values of the parameters, the total storage is $S = S_1 + S_2 = 4106$ words.

For the conventional LPC/DTW recognizer the storage is

$$S_3 = Q \cdot V \cdot T \cdot (p + 1), \tag{20c}$$

which gives 43,200 words for the assumed parameter values. Again we see a 10 to 1 reduction for the HMM/VQ model over the LPC/DTW model.

It should be noted that in our analysis of computation we have not included the computation of LPC coefficients. This computation must be performed in "real-time" and is independent of the vocabulary size, V. Hence, for a sufficiently large vocabulary the computation for scoring each word dominates the overall computation, and the rough analysis given above is appropriate. Furthermore, the computation for coding each LPC vector into the nearest codebook entry [eq. (17)] is also independent of vocabulary size and often could be performed in the "real-time" part of the recognizer. For such implementations the gain in computation of the HMM/VQ recognizer, over the conventional LPC/DTW recognizer, is even higher than our simple analysis predicts. Finally, it is straightforward to show that if we compare the computation of the HMM/VQ recognizer with that of the LPC/DTW/VQ recognizer, assuming that the VQ is done in real time and that tabular computation of products and logarithms is used, then by comparing the number of additions, the computational advantage of the HMM/VQ system over the LPC/DTW/VQ system still holds.

7.2 Some comments on the relationships between DTW and HMMs

Contemporary research on speech recognition has produced two algorithmic procedures for dealing with the nonstationarity of the speech signal: temporal alignment techniques, and Markov modeling. These methods display certain superficial similarities (e.g., both use dynamic programming methods, can be cast in a Bayesian framework, and have a state transition network associated with them), as a result of which it has occasionally been claimed that they are identical. To the best of our knowledge, the experiments reported here represent the first direct comparison of the two methods. From these experiments it is abundantly clear that the methods are not identical. While

* Words of storage refer to unquantized floating point data.

their overall performances are comparable, they appear to make different errors and involve different amounts of computation and different complexities of training.

In all of these respects the two methods reflect the dichotomy between parametric and nonparametric methods of pattern recognition in the sense of Patrick.[23] In Markov modeling we assume that there is a family of models of a particular structure differing only in the values of their parameters. We use a large training set to estimate these parameters and assume that, if correctly done, the parameters will capture the structure of the data. The training procedure is computationally expensive but need be done only once. After training is complete, relatively little computation is required to determine whether an unknown observation was generated by the model.

Temporal alignment methods are opposite in the following ways. We assume that there is an underlying structure to the training data but its form is unknown. We attempt to capture that structure by simply storing one or more samples and measuring their "distance" to an unknown sample with a metric that is sensitive to the distinctive features of the categories that we seek to identify. The metric is monotonically related to the class conditional density functions so that minimum distance corresponds to maximum model probability. In this case training is a computationally simple data collection and storage process. Probability computation, on the other hand, is very costly since we must measure the distance to every prototype in the training set.

All of these characteristics are made manifest by our experiments. What remains to be determined is whether the parity of these methods extends to more difficult problems of speech recognition. We hope to answer that question by further experimentation.

VIII. SUMMARY

We have described the results of an extensive investigation into the applicability of the techniques of vector quantization and hidden Markov modeling to speaker-independent, isolated word recognition. We have shown that, when properly designed, the resulting recognition system produces highly accurate word recognition on a vocabulary of isolated digits. We have also discussed the effects of variations of model parameters on system performance. Our experiments show that the resulting recognizer requires about 10 times less storage, and about 17 times less computation for classifying a test utterance than does an equivalent recognizer using LPC coding and dynamic time warping. These economies are obtained at the expense of only a slight increase in error rate.

REFERENCES

1. T. B. Martin, "Practical Applications of Voice Input to Machines," Proc. IEEE, *64* (April 1976), pp. 487–501.
2. F. Itakura, "Minimum Prediction Residual Principle Applied to Speech Recognition," IEEE Trans. on Acoustics, Speech, and Signal Processing, *ASSP-23*, No. 1 (February 1975), pp. 67–72.
3. L. R. Rabiner and S. E. Levinson, "Isolated Connected Word Recognition—Theory and Selected Applications," IEEE Trans. on Communications, *COM-29*, No. 5 (May 1981), pp. 621–59.
4. N. R. Dixon and T. B. Martin, Eds., *Automatic Speech and Speaker Recognition*, New York: IEEE Press, 1979.
5. W. Lea, Ed., *Trends in Speech Recognition*, Englewood Cliffs, NJ: Prentice Hall, 1980.
6. D. R. Reddy, Ed., *Speech Recognition*, New York: Academic Press, 1974.
7. A. Newell et al., "Speech Understanding Systems," Final Report of a Study Group, Carnegie Mellon Report, May 1971.
8. M. R. Sambur and L. R. Rabiner, "A Speaker-Independent Digit Recognition System," B.S.T.J., *54*, No. 1 (January 1975), pp. 81–102.
9. T. B. Martin, "Acoustic Recognition of a Limited Vocabulary in Continuous Speech," Ph.D Dissertation, Univ. of Penn., 1970.
10. A. B. Poritz, "Linear Predictive Hidden Markov Models and the Speech Signal," Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, 82, Paris, France (May 1982), pp. 1291–4.
11. J. K. Baker, "The Dragon System—An Overview," IEEE Trans. on Acoustics, Speech, and Signal Processing, *ASSP-23*, No. 1 (Feburary 1975), pp. 24–9.
12. F. Jelinek, L. R. Bahl, and R. L. Mercer, "Design of a Linguistic Decoder for the Recognition of Continuous Speech," IEEE Trans. on Information Theory, *IT-21* (May 1975), pp. 250–6.
13. S. E. Levinson, L. R. Rabiner, and M. M. Sondhi, "An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition," B.S.T.J., this issue.
14. L. F. Lamel, L. R. Rabiner, A. E. Rosenberg, and J. G. Wilpon, "An Improved Endpoint Detector for Isolated Word Recognition," IEEE Trans. on Acoustics, Speech, and Signal Processing, *ASSP-29*, No. 4 (August 1981), pp. 777–85.
15. L. R. Rabiner and J. G. Wilpon, "A Simplified Robust Training Procedure for Speaker Trained, Isolated Word Recognition Systems," J. Acoust. Soc. Amer., *68*, No. 5 (November 1980), pp. 1271–6.
16. S. E. Levinson, L. R. Rabiner, A. E. Rosenberg, and J. G. Wilpon, "Iteractive Clustering Techniques for Selecting Speaker Independent Reference Templates for Isolated Word Recognition," IEEE Trans. on Acoustics, Speech, and Signal Processing, *ASSP-27*, No. 2 (April 1979), pp. 134–41.
17. L. R. Rabiner, S. E. Levinson, A. E. Rosenberg, and J. G. Wilpon, "Speaker Independent Recognition of Isolated Words Using Clustering Techniques," IEEE Trans. on Acoustics, Speech, and Signal Processing, *ASSP-27*, No. 4 (August 1979), pp. 336–49.
18. L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains," Annals of Mathematical Statistics, *41*, No. 1 (1970), pp. 164–71.
19. A. Buzo, A. H. Gray, Jr., R. M. Gray, and J. D. Markel, "Speech Coding Based Upon Vector Quantization," IEEE Trans. on Acoustics, Speech and Signal Processing, *ASSP-28*, No. 5 (October 1980), pp. 562–74.
20. A. J. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," IEEE Trans. on Information Theory, *IT-13* (April 1967), pp. 260–9.
21. B. H. Juang, D. Y. Wong, and A. H. Gray, Jr., "Distortion Performance of Vector Quantization for LPC Voice Coding," IEEE Trans. on Acoustics, Speech, and Signal Processing, *ASSP-30*, No. 2 (April 1982), pp. 294–304.
22. M. K. Brown and L. R. Rabiner, "On the Use of Energy in LPC Based Recognition of Isolated Words," B.S.T.J., *61*, No. 10, Part 1 (December 1982), pp. 2971–87.
23. E. A. Patrick, *Fundamentals of Pattern Recognition*, Englewood Cliffs, NJ: Prentice Hall Inc., 1972.