Speaker-Independent Isolated-Digit Recognition Based on Hidden Markov Models and Vocabulary Specific Vector Quantization

by

Louis Cossette

B.A.Sc., Universite de Sherbrooke, 1988

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE (ENGINEERING SCIENCE)

in the School

of

Engineering Science

Simon Fraser University

April, 1991

ISBN 0-315-78304-4

APPROVAL

NAME:	Louis Cossette
DEGREE:	Master of Applied Science (Engineering Science)
TITLE OF THESIS:	Speaker-Independent Isolated-Digit Recognition Based on Hidden Markov Models and Vocabulary Specific Vector Quantization
EXAMINING COMMITTEE:	
Chairman:	Dr. John D. Jones
	Dr. Vladimir Čuperman Senior Supervisor
	Dr. Steve Hardy Supervisor
	Dr. Jacques Vaisey Examiner
DATE APPROVED:	June 27, 1991

ABSTRACT

This thesis presents different systems proposed for speaker-independent isolated-digit recognition. The systems are based on discrete hidden Markov models (HMM). Many of the conventional discrete HMM recognition systems use vector quantization as a data reduction pre-processor. In such a case, vector quantization is often done with a universal codebook. In this thesis, we propose to replace universal codebooks by word-specific codebooks.

The motivation to make this change is based on the fact that VQ can do more than simply quantize the input signal. As a matter of fact, the quantization distortions computed by word-specific codebooks give a distortion score for each word. Moreover, the generation of a set of word-specific VQ index sequences provides a more detailed description of the input signal than if a single index sequence is used.

In our system, word-specific VQs are integrated into the framework of an HMM recognizer. The VQ stage and the HMM stage are connected together with the intention to capitalize on the word-specific VQ distortion scores and index sequences. The results show that word-specific codebooks have some advantages over universal codebooks for isolated-digit recognition systems based on discrete HMMs.

ACKNOWLEDGEMENTS

I would like to thank my supervisor Dr. Vladimir Cuperman for his guidance, his dynamism and his sharp management of my research project. I am also grateful to Dr. Edgar Velez for his invaluable help, for his patience, for the fruitful conversations we had together and for the good suggestions he pointed out to me. I would like to thank Mr. Craig Scratchley who helped me to start correctly my project. I thank Mrs. Brigitte Rabold for her help when I needed information and Mr. Chao Cheng for his assistance on computer related issues. Finally, I would like to thank all my friends in the Engineering department who certainly contributed to the work I have done in SFU. This research project was supported by the Natural Sciences and Engineering Research Council of Canada.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Speech recognition research has evolved considerably in the last few years, but progress is still needed if reliable speech recognizers are to be developed. In order to find applicable solutions, the problem of recognition is often reduced by selecting specific application tasks. In other words, the systems are developed within fixed recognition constraints. For this thesis, the goal is to build a speaker-independent isolated-digit recognizer.

The problem of isolated-digit recognition has been discussed frequently. A main stream in the current research is oriented toward the implementation of hidden Markov models (HMM). The interest for HMMs has been encouraged by the success obtained from HMM-based systems [2] [26]. One of the most successful HMM recognizers is the SPHINX system [12]. In this thesis, the basic techniques of the SPHINX system are applied and adapted to our recognition conditions.

The SPHINX system uses a VQ pre-processor to reduce the amount of data to be processed through the HMMs. However, we will show that VQ can do more than simply quantize a signal for HMM recognition. Information can be extracted from vector quantization and can be used directly for recognition. Different systems will be tested to find out if additional VQ information can improve the performance of HMM-based recognizers.

Since VQ information will be used for recognition, we will focus on the VQ stage. Many characteristics of a vector quantizer can be modified. During this research, I studied how the final results of an isolated-word VQ-HMM recognizer are influenced by the following parameters: the size of the codebooks, the number of codebooks (or VQ sequences), and the individual codebook training set. Also, we will examine how the VQ stage can be connected advantageously to the HMM stage in order to make full use of the informations provided by the VQ stage.

In the next chapter, a general overview of speech recognition is presented to clarify the state of the current research and to place VQ and HMM techniques in their proper context. A system overview will then be presented with the recognition process being divided into three stages: feature extraction, vector quantization, and hidden Markov models. Chapter 3, 4, and 5 explain the basic approach of each processing stage and present the adaptation of these techniques to our specific recognition tasks.

Chapter 6 presents ten different versions of VQ-HMM isolated-digit recognizers. Chapter 7 presents the experimental results obtained by the systems proposed in Chapter 6. Some conclusions are drawn from every recognizer's performances. Finally, a general conclusion closes the discussion.

CHAPTER 2

SPEECH RECOGNITION OVERVIEW

2.1 Constraints in speech recognition

The ultimate goal of speech recognition is to build a recognizer working under any possible conditions, without any restrictions. However, the diversity of speech signals and words is so wide that some constraints need to be assumed. Therefore, it is first necessary to define the constraints of the recognizer. The possible constraints are : 1) speaker dependence, 2) connectivity between adjacent words, 3) vocabulary size.

1. Speaker-dependence or speaker-independence

   A speaker-dependent recognizer is a system trained to be used by a limited number of speakers. A speaker-independent system, however, is capable of recognizing speech from any new speaker. Since speaker-independence implies a much wider variety of speakers, it is a significantly more difficult task than speaker-dependent recognition. The system considered in this thesis is speaker-independent.

2. Isolated words or continuous speech

   Ideally, a speech recognition system should recognize continuous speech in the form of normal conversation, without any pauses enforced between the words. However, continuous speech recognition brings many problems for three main reasons.

   First of all, word boundaries in continuous speech are difficult to define and to locate. Consequently, searching becomes more complex and recognition less accurate than in the isolated-word case. A second difficulty that arises for continuous speech is the presence of co-articulatory effects. Since the articulators of the vocal tract don't move instantaneously from one position to the other, the phonemes of a utterance can be strongly influenced by neighboring phonemes. For continuous speech recognition, the co-articulatory effects are very hard to predict. Thirdly, short words such as articles, prepositions, pronouns, or short verbs, are often shortened, skipped, distorted or simply poorly articulated.

   Continuous speech recognition is thus a difficult problem and viable implementations require high computational complexity. The difficulties associated to continuous speech require solutions going beyond the scope of this thesis. Therefore, we will focus on isolated-word recognition where each word is isolated by a definite pause. Word-spotting will not be necessary and the problems of co-articulatory effects will be easier to handle since the number of possible combination of neighboring phonemes is reduced. However, many problems remain, especially if the input signal contains noise.

3. Vocabulary size

   Generally, small vocabularies are easier to recognize than large ones because there is less possible confusion and the complexity of search is lower. With small vocabularies, each word can be modeled separately as a distinct reference. But for large vocabularies (1000 words and more), it is normally not possible to train a separate model for each word because of the limits imposed for training time and storage space. Instead of using word models, it is possible to create subword models. Subword units can be phonemes, broad phonetic categories, diphones, demi-syllables, syllables, etc. For large vocabularies, there is normally much less subword units than words. In the English language, there are up to 100000 words but only about 1000 demi-syllables, 2500 diphones , 20000 syllables and less than 100 phonemes [12]. The subword units can eventually be combined at higher levels to form the desired utterances.

   The choice of subword units is critical and can have a determining influence on the performance. The choice of unit is determined by the number of units present in the vocabulary and on the system's capability to discriminate between the descriptive features of the different units. The units must also be insensitive enough to context. Finally, the hierarchical structure through which sequences of units are combined into words must be general enough to make the reconstruction of utterances possible.

The vocabulary used in this thesis is limited to the single digits. So, with such a small vocabulary, it is possible to implement word models instead of subword models.

Once the constraints are established, the speech recognition system has to be designed. The next section describes the most generally used speech recognition techniques.

2.2 Literature review

2.2.1 Speech recognition model

Speech recognition is a general problem that leads to many different proposed solutions and systems. Although the existing recognizers may use very different techniques, they are built following general rules defined by pattern-recognition models. Fig. 2.1 shows a pattern-recognition model used for isolated-word recognition [28]. This section, as well as this thesis, is limited to the problem of isolated-word recognition. Nevertheless, the basic techniques developed for isolated-word recognition are often similar to those which are applied for continuous speech recognition.

The three basic stages of the model shown in Fig. 2.1 are:

1. feature extraction;

2. pattern similarity measurement;

3. decision rule.

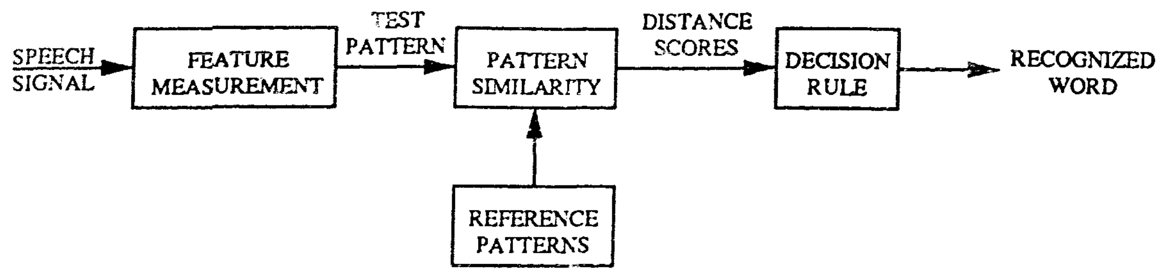


Figure 2.1: Block diagram of a pattern-recognition model

The input to the system is the speech waveform itself. The signal is digitized and transformed into feature parameters which should preserve the relevant information of the input signal. Many different features can be extracted and the choice available depends on the model to be implemented [28]. The computation time, the storage space and the ease of implementation also are important factors to consider when sets of features are chosen.

Once the input features have been processed, they are compared to reference patterns which are separated into classes. For example, different words or different phonemes could be separate classes. Different classes can also be created for different features. Training procedures form reference patterns for each class by averaging many features of a same class. For recognition, the system identifies the reference patterns that match the input features the most closely.

The three steps of the pattern-recognition model are interdependent. The pattern similarity measurement depends on the feature extracted. The decision rule is also determined by the set of distortion scores obtained from the pattern similarity block. Consequently, the design of a speech recognizer is not straightforward, all design decisions are related to each other and have an impact on the final system performance.

2.2.2 Feature measurement

Feature measurement is basically a data reduction process whereby a large number of data points (samples of the speech waveform) are transformed into a smaller set of features which are relevant in the sense that they contain the important properties of the acoustic waveform. It would be computationally costly to use directly the speech samples to make pattern similarity measurements. In this section, the main features used for speech recognition are presented.

Amplitude or power versus time

The main reason to use power as a feature is to differentiate between silence and speech or between voiced and voiceless sounds. It can also be used to detect word endpoint information. Zero-crossing information can be used as a complement to energy to identify the fricatives and sibilants [23]. The technique used to calculate power will be described in the next chapter.

Filter bank analysis

Filter-bank analysis [27] provides a computationally cheap and fast way to extract spectral information of a signal. The basic idea is to pass the speech signal through different bandpass filters. The filters are put on parallel branches and extract different adjacent band frequencies. The bank of filters covers normally the frequencies from 100Hz to 3000Hz or 8000Hz. The number of filters can vary from about 5 to as many as 32, and the filter spacing is generally linear until about 1000Hz and logarithmic beyond 1000Hz. Nonuniform spacing of the filters is used to exploit the ear's decreasing frequency resolution with increasing frequency.

The output of each bandpass filter is usually passed through a non-linear element and a low-pass filter to give a signal which is proportional to the energy of the speech signal in the band. The resulting signal coming out of each branch is proportional to the energy of the input signal in that particular band. The outputs of the branches are used as the features for the recognizer.

Fourier analysis

Fourier representation has traditionally played a major role in speech processing. Fourier representations give good descriptions of the spectrum of a signal. Standard Fourier representations are usually appropriate for periodic, or stationary random signals. Speech is a non-stationary signal. However, temporal properties such as energy and correlation are usually assumed to be fixed over short time intervals (10 ms to 30 ms). Therefore, it is possible to do Fourier analysis by applying the short-time Fourier transform [27] on such intervals.

LPC parameters

The linear predictive coding (LPC) technique [18] [1] has been found to be a robust, reliable and accurate method to estimate the characteristics of linear, time varying systems. The speech production can be modeled as the output of a linear, time varying system excited by periodic pulses or random noise. Consequently, LPC is widely used for the purpose of speech recognition. A detailed description of LPC analysis will be done in Section 3.7.

Cepstrum

A voice signal has a rapidly-varying component (vocal cord excitation) and a slowly-varying component (vocal tract changing). The magnitude spectrum $|X(\omega)|$, $X(\omega)$ being the Fourier transform of the signal, gives us information about the distribution over the frequencies at an instant t. However, it is not possible to extract, directly from $|X(\omega)|$, information about the rapidly and slowly varying components of the signal.

However, this task can be accomplished by calculating the cepstrum of the signal [21] which is defined as $F^{-1}(\log|X(\omega)|)$. The cepstrum is an homomorphic deconvolution [27] of the input signal. As it will be seen in the LPC analysis (3.7), voiced speech can be modeled as a train of pulses (vocal cord excitation) passed through a linear filter (vocal tract representation). Therefore, the speech signal can be seen as a result of a convolution. The cepstrum transforms first this convolution into a multiplication by using the Fourier transform and finally reduces the signal to an addition by applying a logarithm on the spectrum. As a result, the spectral envelope and the fundamental period can be separated by homomorphic deconvolution.

Formants

The vocal tract, like any other acoustic tube, has natural frequencies which are a function of its shape. These natural resonances are called formants and are the most important acoustical characteristics of the vocal tract. Formants represent listener's primary source of information about the position of the speaker's vocal organs and they are identified by number in order of increasing frequency: F_1, F_2, etc. [23] In speech recognition systems, at least the three first formants are considered.

Informations about formants is contained in the spectral envelope. Hence, all formant estimators either implicitly or explicitly examine the spectrum envelope. In many procedures, the maxima of the envelope are considered to be the formants.

Warping scales

Spectral features can be warped in frequency in order to simulate the auditory characteristics of the human ear. The resolution of the human ear at high frequencies is less sharp than at low frequencies. Moreover, formant bandwidth increases with frequency. Many scales have been proposed to simulate this aspect of hearing perception.

The Bark scale [8] corresponds to the frequency scale on the basilar of the ear's cochlea. It is defined as:

$$B = 13\arctan(0.76f) + 3.5\arctan(\frac{f}{7.5})^2 \tag{2.1}$$

where B and f represent the Bark scale and frequency in KHz.

The Mel scale [8] corresponds to the auditory sensation of tone height. The scale is given by:

$$Mel = 1000 \log_2(1 + f) \tag{2.2}$$

The bilinear transform [12] warps the linear axis using an all-pass filter. This scale is comparable to the Bark scale or the Mel scale when the filter element a takes a value between 0.4 and 0.8. The filter is defined as:

$$z_{new}^{-1} = \frac{(z^{-1} - a)}{(1 - az^{-1})}, (-1 < a < 1) \tag{2.3}$$

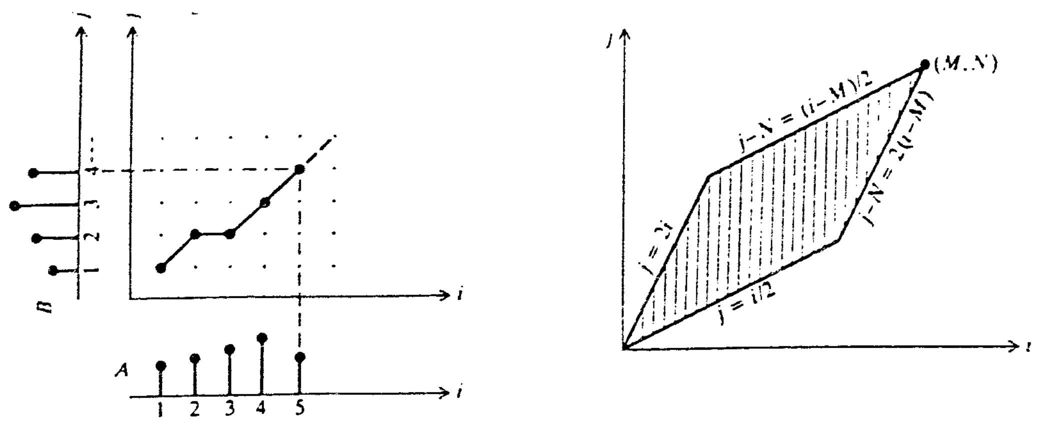
$$\omega_{new} = \omega + 2\tan^{-1}(\frac{a \sin \omega}{1 - a \cos \omega}) \tag{2.4}$$

where ω is the sampling frequency expressed by the normalized angular frequency, ω_{new} is the converted frequency, and a is the frequency warping parameter. A positive a converts the frequency axis into a low-frequency weighted scale by lengthening the low-frequency axis and shortening the pre-frequency axis.

2.2.3 Pattern similarity measurement and decision rule

Once the features of the input signal are extracted, they are matched to the reference patterns. A strategy is elaborated to measure the similarity between the input and the reference patterns.

Three classes of recognition systems are presented in this section: the template matching, the probabilistic modeling and the knowledge-based approach.



(a) (b)

Figure 2.2: Time warping process (a) and path constraints area (b)

Template matching

The template matching method is a decision making process matching the input signal to each of a set of templates. In many isolated-word recognition systems, the reference patterns are calculated from features recorded over the length of the whole word rather than at particular points; such patterns are called templates. For example, in vector quantization, the templates are represented by codevectors. We will cover vector quantization in Chapter 4. In this section, the discussion will be restricted to dynamic time warping alignment (DTW).

Dynamic time warping solves the problem of aligning the input feature signals and the feature templates when the phonetic events are not consistent in time and when the durations are different. Time warping is the process through which the time axis of the input is nonuniformly distorted, or warped, to align it with the time axis of the reference pattern. This process is illustrated in Fig. 2.2(a) [23].

The best time alignment path is a curve relating the j time axis of the reference pattern to the i axis of the input signal. A distance measure is calculated when the signals are compared at the position defined by the coordinate of the points. The distortion measure is accumulated along the points of the paths. The path that generates the smallest distortion defines the best warping. Since there is a large number of possible paths to test, constraints are often imposed to limit the number of computations:

1. Endpoint constraints on the path.

   Normally the path starts at (1,1) and ends at the top right corner (M,N) of the grid.

2. Local path constraints.

   The possible types of motions (e.g., directions, slopes) from one point to another are restricted.

3. Global path constraints.

   The path can fall only in a chosen area of the (i,j) plan. The shadowed surface of Fig. 2.2(b) is a possible global path region. The parallelogram imposes reasonable limits to the search.

4. Distance measure.

   The distance measure is used to search the optimal paths and to eliminate the paths with high accumulated distortions.

Several distance measures exist and depend on the feature sets. A list of the most frequently used distance measures follows:

- The Euclidean distance is defined as:

$$d(a_i, b_j) = \sum_{k=1}^{K}(a_i(k) - b_j(k))^2 \tag{2.5}$$

where a_i and b_j are the signals at i and j on the respective time axis. Euclidean distance will be the distance measurement that will be used in the process of vector quantization for this thesis.

- The covariance weighting is used to compensate for correlation between features and it tends to give equal weight to all features for the overall distance calculation. The covariance weighting measure is defined by:

$$d(a_i, b_j) = (a_i - b_j)\tau^{-1}(a_i - b_j)^T \tag{2.6}$$

where τ^{-1} is the inverse covariance matrix of the features.

- The log spectras distance can be calculated with the following formula:

$$d(a_i, b_j) = \int_{\omega}[\log(a_i(e^{j\omega})) - \log(b_j(e^{j\omega}))]^q d\omega \tag{2.7}$$

The integral covers a chosen frequency range and q is an even integer that makes positive the power of the difference.

- The Itakura's log likelihood measure proved to be efficient to measure distances between LPC derived features. The equation is:

$$d(a_i, b_j) = \log[\frac{a_i R a_i^T}{b_j R b_j^T}] \tag{2.8}$$

where a_i and b_j are the LPC coefficients of the signals and R is the autocorrelation matrix of the input signal.

Even under the constraints (1 to 4) mentioned previously, the number of possible paths to evaluate remains prohibitive. Dynamic programming for time warping (Dynamic time warping) solves that problem and achieves reasonable computational complexity. This procedure is based on the fact that the best path from (1,1) to any given point is independent of what happens beyond that point. Consequently, the accumulated distance D_A is:

$$D_A(i,j) = d(a_i, b_j) + \min_{q \leq j}(D_A(i-1, q)) \tag{2.9}$$

where d is the local distance between feature sets a_i and b_j.

With dynamic programming, the distance for every possible next steps is calculated for each path. At each point, the best predecessor (the least cost path) is chosen. A search can lead to many ramifications, but the number of ramifications is limited by the set of constraints. A maximum permissible accumulated distortion D_A can also be imposed. Once the point (M,N) is reached, backtracking is done to retrieve the best path.

DTW has shown very good recognition rate [34] but it still requires a fair amount of computations. Consequently, other methods often replace DTW for speech recognition applications.

Hidden Markov Modeling

Hidden Markov modeling (HMM) is a recognition strategy based on searching through stochastic models. HMM offers comparable performances to those of DTW in many applications at a fraction (up to 17 times less [29]) of the computational cost. HMM is the basic stage of the speech recognizer proposed in this thesis and a complete coverage of this technique is done in Chapter 5.

Knowledge-based systems

To understand conversations, it is known that humans make use of their extensive knowledge of speech and their ability to predict the next words. Taking this into account, we can assume that recognition is not only done by using human perceptual and analytical powers but also by using a knowledge of the language. As a result, knowledge-based recognition systems were developed using the knowledge that humans have about the language.

The ARPA research agency was set up to encourage research for knowledge-based systems and some encouraging results have been obtained [17] but not yet comparable to those obtained with DTW or HMM.

2.3 Proposed isolated-digit speech recognizer

The discussion throughout this chapter concentrated on the general aspects of speech recognition systems. Different techniques have been presented to give an idea of past and current research. We now focus on the systems that are to be proposed in this thesis.

Our recognizers are based on hidden Markov modeling which is one of the most popular recognition methods of the last few years. The baseline speech recognizer developed in this project is based on approaches developed for SPHINX [12], one of the most successful recent recognizers. However, many modifications were done

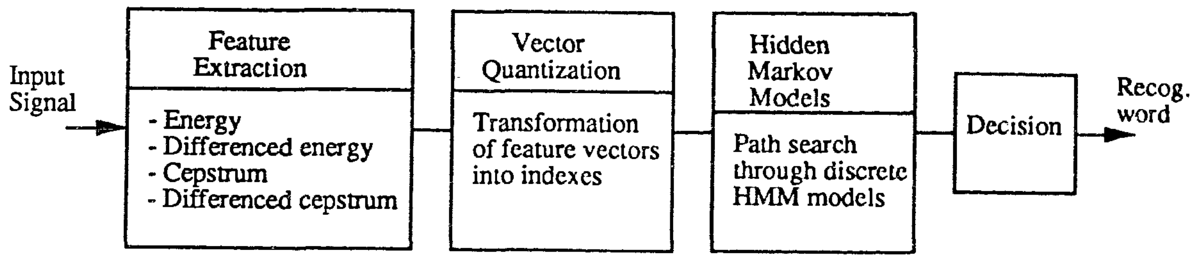


Figure 2.3: Schematic diagram of the proposed recognizer.

since our recognition requirements are different from the objectives of the SPHINX system and we tried to improve recognition performance for the particular task of small-dictionary isolated word recognition.

The proposed recognizer can be subdivided into three stages: feature extraction, vector quantization and hidden Markov models. Fig. 2.3 shows the block diagram of the system and the function of each stage. The three processing stages are presented separately in the next three chapters.

CHAPTER 3

FEATURE EXTRACTION

The first stage of the proposed speech recognition system is the feature extraction stage. The input signal is digitized and compressed into three feature vectors: combined energy and differenced energy; cepstrum coefficients; and differenced cepstrum coefficients. These features are commonly used in speech recognition [12], cepstrum notably performs well because it gives directly an accurate smoothed estimate of the spectral envelope of the signal. LPC-derived cepstrum was preferred to other LPC-based features because it has shown superior performance for speech recognition [30] [37]. This chapter describes how the features are calculated from the speech signal. A block diagram of the feature extraction process is shown in Fig. 3.1.

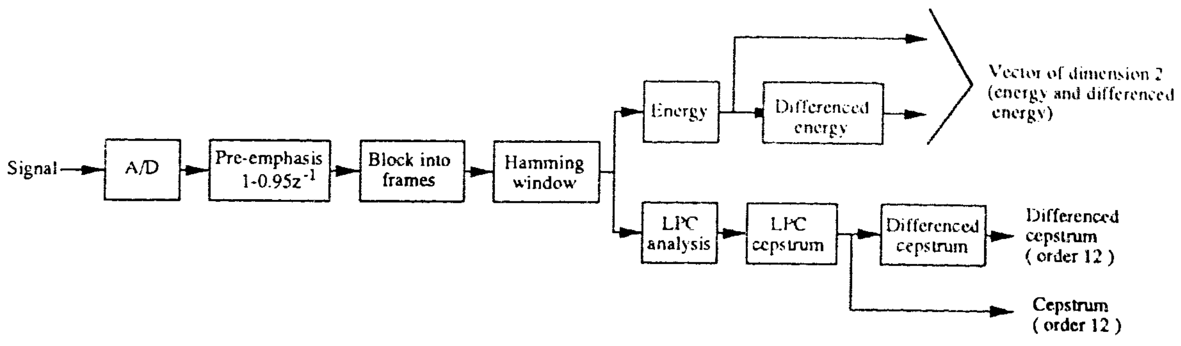


Figure 3.1: Block diagram of the feature extraction stage.

3.1 Digitization

Two databases were used to conduct the experiments in this thesis. The first database was recorded in a studio and the second database was recorded over telephone lines.

The tokens (a token is a speech segment) of the studio database were first filtered by an anti-alias filter with a cutoff frequency at around 8 KHz. The speech was then sampled at a rate of 32 KHz with 12 quantization bits. A digital filter with a 3.4 KHz cutoff was applied to the sampled speech. Finally, the digital signal was down sampled to 8 KHz by extracting one sample at every 4 samples.

The telephone database was filtered through an anti-alias filter with a cutoff frequency at 4.8 KHz. The sampling frequency was set to 10 KHz and the signal was quantized on 12 bits.

3.2 Pre-emphasis

To reduce the possibility of computational instabilities due to finite precision arithmetic, the signal for each utterance is spectrally flattened by pre-emphasis. The filter used for pre-emphasis is :

$$H(z) = 1 - 0.95z^{-1} \tag{3.1}$$

3.3 Block into frames

The digital signal is blocked into frames of 20 ms. For the studio database, the frames contain 160 samples and for the telephone database, the frames contain 200 samples. In both cases, the consecutive frames are overlapped by 10 ms.

3.4 Hamming window

To reduce spectral spreading due to the Gibbs phenomenon, each frame is multiplied by a Hamming window. The multiplication of the speech wave by the window function gradually attenuates the amplitude at both ends of the extraction interval to prevent an abrupt change at the endpoints. The equation for a Hamming window is :

$$w(n) = 0.54 - 0.46cos(2\pi n/(N-1)) \tag{3.2}$$

3.5 Energy

We incorporate energy as a feature in our recognition system. Energy is mainly used because it facilitates the separation of speech from silence. Energy can simply be computed from the waveform with:

$$E_t = \log(\sum_{i=1}^{M} x_i^2) \tag{3.3}$$

where E_t is the energy for frame t, which has M discrete time samples in it, namely $x_1, x_2, \cdots, x_M$.

The absolute value of the energy can be used to detect silence by fixing an energy threshold under which the signal is considered to be silent. However, the absolute energy is not a reliable source of information in speech recognition because the voice loudness for two speakers may be quite different. Consequently, the input tokens need to be normalized. Normalization is done by subtracting the peak energy value of a token from the energy of all the other frames of the token:

$$E_{norm} = E_t - E_{max} \tag{3.4}$$

E_t is the energy value for frame t and E_{max} is the overall maximum frame energy for the token considered.

Normalization cannot be applied without any consideration on the nature of the token (the token can be a word or a phoneme). The normalization of a phoneme is not comparable to the normalization of a word. For example, the normalized energy for a low energy phoneme within a word (using the peak energy of the word) may give much lower normalized values than if this low energy segment was normalized as a phoneme (using the phoneme's peak energy). Therefore, it would be meaningless to compare normalized energy values of an input word if the energy values of the reference templates have been normalized from the peak energy of phoneme tokens.

A way to solve the above problem is to normalize the energy of a phoneme by using the peak energy of the word from which that phoneme was segmented.

3.6 Differenced energy

The differenced energy [12] is calculated in order to estimate the slope of the energy signal at different times, which can be helpful to recognize the plosives since the slope locates the changes in loudness of the signal. The differenced value for frame t is obtained by making the difference between the energy of frame t+2 and t-2:

$$\Delta E_t = E_{t+2} - E_{t-2} \tag{3.5}$$

3.7 LPC analysis

Linear predictive coding (LPC) [18] [1] is a basic technique for estimating speech parameters with linear combinations of past speech samples. The LPC coefficients calculated in the LPC analysis also provide accurate estimates used for the modeling of the vocal tract and are thus used as features in our system. The next paragraph gives a model for speech production. From this model, we will see how LPC coefficients can be used to characterize the speech signal.

The production of speech can be modeled as the output of a linear, time-varying system excited by either periodic pulses (during voiced speech), or random noise (during unvoiced speech). The transfer function between the input (excitation) and the output (speech signal) can be approximated by:

$$H(z) = \frac{G}{(1 - \sum_{k=1}^{M} a_k z^{-k})} \tag{3.6}$$

The speech samples $(y_i(n))$ are then related to the excitation $(x_i(n))$ by an all-pole filter model:

$$y_i(n) = \sum_{k=1}^{M} a_k y_i(n-k) + G x_i(n) \tag{3.7}$$

In this model, G is the gain parameter, the a_k's are the filter coefficients and M is the order of the analysis.

On the other hand, from the linear prediction theory, we know that $y_i(n)$ can be predicted with:

$$\hat{y}_i(n) = \sum_{k=1}^{M} \alpha_k y_i(n-k) \tag{3.8}$$

And, the prediction error, $e(n)$, is defined as:

$$e(n) = y_i(n) - \hat{y}_i(n) = y_i(n) - \sum_{k=1}^{M} \alpha_k y_i(n-k) \tag{3.9}$$

From Equation 3.9, it can be seen that the prediction error sequence is the output of a system whose transfer function is:

$$A(z) = 1 - \sum_{k=1}^{M} \alpha_k z^{-k} \tag{3.10}$$

By comparing Equations 3.7 and 3.9, we realize that if $\alpha_k = a_k$, then $e(n) = G x_i(n)$. Thus, the prediction error filter, $A(z)$, will be an inverse filter for the system, $H(z)$, of Equation 3.6:

$$H(z) = \frac{G}{A(z)} \tag{3.11}$$

From these models, it is showed in [27] that the LPC coefficients (α_k) give good estimates of the spectral properties of the speech signal. We can see from Equation 3.9 that the minimization of the prediction error allows to find a set of α_k's. But, because of the time-varying nature of the speech signals, the predictor coefficients must be re-estimated for every short segments of the speech signal. If the signal is divided into frames of length N, the resulting prediction error for a Hamming windowed frame is:

$$E_i = \sum_{m=0}^{N+M+1} [y_i(m) - \sum_{k=1}^{M} \alpha_k y_i(m-k)]^2 \qquad (3.12)$$

The minimization of the mean-squared error is achieved using the autocorrelation method [18] [19] because in our system's conditions (frame size equal to 200 samples and LPC order 14), it was shown that the autocorrelation method was as good as the covariance method and superior to the lattice method [27]. The details of the autocorrelation method are given in [27] and are summarized in the following equations.

The expression of a short-time autocorrelation function of the interval $0 \leq m \leq N-1$ is defined with:

$$r_i(j) = \frac{1}{N} \sum_{m=0}^{N-1-j} y_i(m) y_i(m+j) \qquad 0 \leq j \leq M \qquad (3.13)$$

Given this definition, the minimization of the mean squared error (3.12) leads to the following expression:

$$r_i(j) = \sum_{k=1}^{M} \alpha_k r_i(|j-k|) \qquad 1 \leq j \leq M \qquad (3.14)$$

This is a system of M equations and M unknowns, $\alpha_1, \alpha_2, \cdots, \alpha_M$, which can be expressed in a matrix form as:

$$\begin{bmatrix} r_i(0) & r_i(1) & \cdots & r_i(M-1) \\ r_i(1) & r_i(0) & \cdots & r_i(M-2) \\ r_i(2) & r_i(1) & \cdots & r_i(M-3) \\ \vdots & \vdots & & \vdots \\ r_i(M-1) & r_i(M-2) & \cdots & r_i(0) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \vdots \\ \alpha_M \end{bmatrix} = \begin{bmatrix} r_i(1) \\ r_i(2) \\ r_i(3) \\ \vdots \\ r_i(M) \end{bmatrix}$$

The $M \times M$ matrix is a Toeplitz matrix, i. e. , it is symmetric and all the elements on any diagonal (from up left to down right) are equal. This matrix can be efficiently solved with Durbin's recursive solution [18]. The resulting information obtained from LPC analysis is a set of M LPC coefficients extracted for each frame of the input signal. The order M of the analysis was set to 14. According to [27], we need 3 or 4 poles to represent the source excitation spectrum and we need one pole per KHz (of the sampling frequency) to represent the contribution of the vocal tract to the speech spectrum. In our case, the sampling frequency was 10 KHz (8KHz for the studio database) and that is why order 14 was chosen for the LPC analysis.

3.8 Cepstrum coefficients

The cepstrum [21] is a function from which it is possible to extract separately the spectral envelope of a signal and its fundamental period. As it was defined in Section 2.2.2, the cepstrum can be obtained by computing the inverse Fourier transform of the log Fourier transform of the signal.

The cepstrum can also be evaluated with a method based on the homomorphic analysis [8] (see Section 2.2.2). As a result, the cepstrum coefficients can be recursively calculated from the LPC coefficients:

$$
\begin{aligned}
C_1 &= -\alpha_1 && && (3.15)\\
C_n &= -\alpha_n - \sum_{m=1}^{n-1}(1-\frac{m}{n})\alpha_m C_{n-m} && (1 < n \leq p) && (3.16)\\
C_n &= -\sum_{m=1}^{p}(1-\frac{m}{n})\alpha_m C_{n-m} && (p < n) && (3.17)
\end{aligned}
$$

where α_i's are linear prediction coefficients and C_i's are an LPC cepstrum coefficients. Like in the case of the SPHINX system [12], the order of the LPC analysis (p) is fixed to 14 and the dimension of the cepstrum vectors (n) is fixed to 12 coefficients. These dimensions correspond approximately to those generally used for speech recognition systems. The cepstrum vector dimension must be high enough to give good estimates of the cepstrum and low enough to avoid cumbersome computations.

3.9 Differenced cepstrum coefficients

Temporal changes in the spectra are believed to play an important role in human perception. Therefore, differenced coefficients [12] are used to capture the slope of the temporal changes. The differenced cepstrum coefficients are calculated using:

$$\Delta C_t(k) = C_{t+2}(k) - C_{t-2}(k) \qquad (3.18)$$

where $C_t(k)$ is the kth element of the cepstrum at frame t.

The slope is calculated from a difference of 4 overlapped frames (40 ms like in the SPHINX system). For the first two frames of a utterance, it is impossible to get the term $C_{t-2}(k)$. Consequently, the slope is calculated by doing an extrapolation:

$$\Delta C_t(k) = \frac{4}{t_f - t_i}(C_{t_f}(k) - C_{t_i}(k)) \qquad t_f - t_i < 4 \qquad (3.19)$$

where t_f and t_i are the ending frame and the beginning frame of the slope. The same interpolation is done for the last two differenced coefficients of the utterance.

3.10 Feature vectors

The resulting features obtained from the process illustrated on Fig. 3.1 are three vectors generated frame by frame:

1. Energy vector of dimension 2. The first element is the differenced energy of a frame and the second element is the energy itself.
2. Cepstrum coefficients as a vector of dimension 12.
3. Differenced cepstrum coefficients as a vector of dimension 12.

CHAPTER 4

VECTOR QUANTIZATION

Vector quantization(VQ) is a data reduction technique that maps a real vector onto a discrete symbol. Vector quantization is used to compress input signal feature vectors into single indexes. At each frame, 24 feature vector elements are compressed into 3 indexes. Consequently, the amount of computations for the following stages of the proposed system is greatly reduced. Despite the compression of the information, vector quantization results in only a little lost in accuracy [30].

The basic concept of vector quantization is schematically depicted in Fig. 4.1. A vector quantizer is completely described by a codebook and a distortion measure. A codebook is a finite collection of vectors called codevectors $(C_n(k))$. Each codevector has the same dimension as the input vector $(V(k))$. The codebook is trained to represent the distribution of the training vectors, and to minimize the total distortion of each training vector against the best matching codevector.

In our system, the codebooks are trained with the LBG algorithm [16]. The

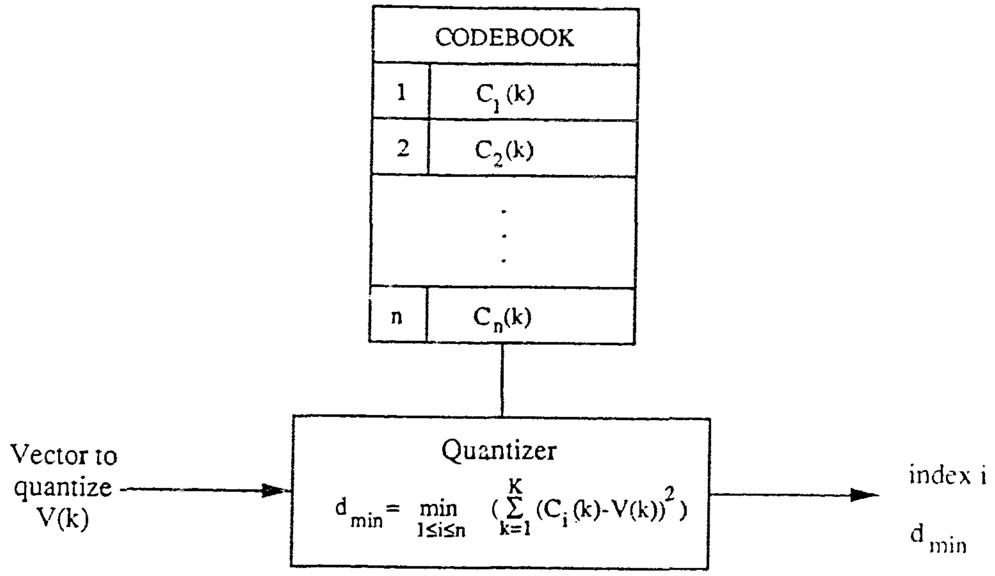


Figure 4.1: Schematic diagram of a vector quantizer.

LBG algorithm is a clustering algorithm, used to separate the training data set into groups, or clusters, of similar data items and each cluster's centroid is assigned as a codebook codevector.

The VQ output is the index of the codeword which best matches the input vector. The VQ can also output a distortion measure (d_{min}) giving the distance between the input vector and the chosen codevector. As we will see later (Chapter 6), this distortion measure can be useful for the recognition decision procedure.

The next sections describe in more detail the VQ training procedure and the method used to generate VQ indexes.

4.1 Codebook training

The goal of a vector quantization training algorithm is to generate a number of codevectors from a large sample of training vectors. Codebook training selects the codevectors representing the distribution of the training vectors and minimizing the total distortion of each training vector against the best matching codevector. In this section, the distortion criterion and the codebook training algorithm are presented.

4.1.1 Distance measurement

For the training procedure and the quantization process of our VQ stage, the distortion measure used to calculate the distance between two vectors is the Euclidean distance. Many speech recognizers, including the SPHINX system [12], use the Euclidean distance measurement because it gives good comparison scores for cepstrum coefficient vectors [37]. The distortion measure is given by:

$$d(V_1, V_2) = \sum_{m=1}^{M} (V_1(m) - V_2(m))^2 \tag{4.1}$$

where V_1 and V_2 are the two vectors to be compared.

4.1.2 LBG algorithm

The LBG algorithm [16] is a VQ clustering process splitting consecutively the training data into $2, 4, 8, \cdots, 2^B$ clusters. The training set is divided into clusters by iterative refinement. Each training vector is classified into the cluster whose centroid best matches the vector. Once all the training vectors have been classified, a new centroid is calculated for each cluster (or cell) by averaging all the vectors of the cluster. This process is repeated a fixed number of times to reduce the average distortion (the average distance between the vectors and the centroid) of each cluster. The steps of a bit stage are illustrated in the flowchart of Fig. 4.2.

At the beginning of the training algorithm, we need to choose a set of initial centroids. In the case of the K-means algorithm, the number of initial centroids is equal to the size of the codebook to be trained. This method limits the training procedure to one bit-stage, however, some codevectors may be poorly trained due to the fact that the values of the initial centroids may not be properly set. In our case, we avoid this problem by starting with two initial clusters from which the initial centroids are calculated. And, when the centroids of the clusters have converged, the clusters are split by separating each current centroids into two new centroids. The old centroids are simply multiplied by pre-determined factors. The iterative process is then applied to the new centroids. The algorithm ends when the number of centroids is equal to the number of codevectors needed to be trained for the VQ codebooks. Each centroid is stored as a codevector.

The LBG algorithm can be summarized by the following steps:

1. Start with two initial codevectors.

2. For each vector in the training set of a codebook, do a full search over all available codevectors to find the nearest neighbor (using Eq. 4.1) and then assign the input to the corresponding cell.

3. Update the centroid for each cell by computing the average of the vectors of the cell and use the new centroids as the current codevectors.

4. Repeat 2. and 3. a fixed number (N) of times.

5. If the current number of codevectors equals L (number of codevectors to train for the codebook), repeat 2. and 3. for a fixed number (M) of times more, and then stop. Steps 2. and 3. are repeated M times to be sure that the final centroids have converged.

6. If the current number of codevectors is inferior to L, initialize codevectors for the next bit stage by splitting each centroid into two: one equal to the old centroid, and the other equal to the old centroid multiplied by a factor (in our case 1.01). Then, go to 2.

For some applications, the size of the codebooks (L) needs to be set to a number which is not a power of 2. Therefore, the codebook training procedure has to be altered. The following operations are added to generate codebooks of any size:

1. Train a codebook of size 2^b where $2^b < L < 2^{b+1}$.

2. Choose the $(L - 2^b)$ centroids having the highest average cluster distortion.

3. Split the $(L - 2^b)$ chosen centroids to obtain, with the first 2^b centroids, a total of L centroids.

4. Train the L centroids with the clustering algorithm to obtain the final codebook.

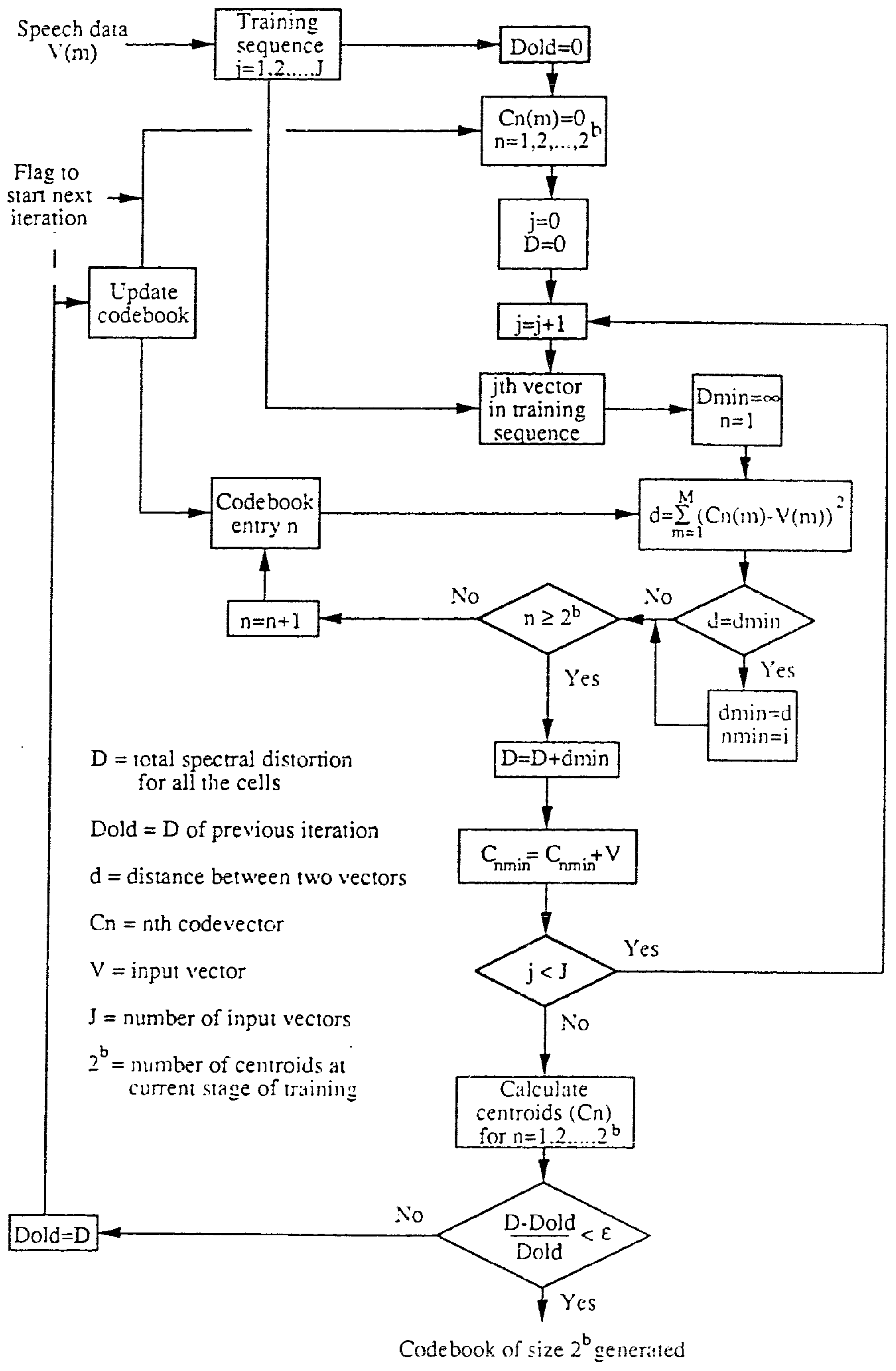

Speech data V(m)
Training sequence j=1,2,....J
Dold=0
Cn(m)=0 n=1,2,...,2^b
Flag to start next iteration
j=0 D=0
j=j+1
Update codebook
jth vector in training sequence
Dmin=∞ n=1
Codebook entry n
d=Σ_{m=1}^{M} (Cn(m)-V(m))^2
No
n ≥ 2^b
No
d=dmin
Yes
n=n+1
Yes
dmin=d nmin=i
D=D+dmin
D = total spectral distortion for all the cells
Dold = D of previous iteration
d = distance between two vectors
Cn = nth codevector
V = input vector
J = number of input vectors
2^b = number of centroids at current stage of training
C_nmin = C_nmin + V
Yes
j < J
No
Calculate centroids (Cn) for n=1,2,....2^b
No
Dold=D
(D-Dold)/Dold < ε
Yes
Codebook of size 2^b generated

Figure 4.2: Iteration procedure for codebook generation when codebook size is 2^b

4.2 Vector quantization process

4.2.1 Output index and distortion score

Once the codebooks have been trained, vector quantization is done by matching one by one the input vectors against the codevectors. Equation 4.1 is used as the vector comparison distortion measure. The index of the closest codevector to the input vector is output from the VQ (Fig. 4.1):

$$d_{min} = \min_{1 \le n \le N} [\sum_{m=1}^{M} (C_n(m) - V(m))^2] \tag{4.2}$$

$C_n(m)$ is the mth coefficient of the nth codevector in the codebook and $V(m)$ is the mth coefficient of the input vector.

d_{min} is the distortion resulting from the quantization of one input vector. The quantization distortion can be accumulated for all the frames of an input utterance. A final distortion score can be obtained for each codebook by averaging the d_{min} of all the utterance's input vectors:

$$D_i = (\sum_{t=1}^{T} d_{min,j})/T \tag{4.3}$$

D_i is the distortion score for codebook i and T is the number of frames in the input token. The distortion score will be used as a VQ information for the recognition decision in some of our systems (see Chapter 6).

4.2.2 Multiple codebooks

It has been seen in the chapter on feature extraction, that three sets of features are generated. At every frame, one cepstrum vector, one differenced cepstrum vector and one energy vector are calculated. It is necessary to define the process by which the feature vectors are processed by the VQ stage.

The approach suggested in this thesis is the multiple codebook vector quantization [9]. Lower quantization distortions are obtained when the feature vectors are partitioned in different feature codebooks rather than used in a common codebook where a codevector would represent the three features in a same vector [12]. A different set of codebooks is trained for each feature. Consequently, for each frame of speech, not one but three VQ indexes are used to describe the input signal.

One problem with the multiple codebook approach, however, is the need of substantially more storage. In our case, the number of parameters for the models is tripled.

CHAPTER 5

HIDDEN MARKOV MODELS

At this point in the system's description, temporal information was analyzed only in the feature extraction stage when the differenced coefficients were calculated. Differenced coefficients represent the slope of a signal at different times. It is evident that this information gives only a part of the temporal information in a signal. In the chapter on VQ, we ignored the discussion on temporal variations of a speech signal because the VQ algorithm used is unable to capture this type of information.

In this chapter, the probabilistic theory of hidden Markov chains is introduced as a solution for the modeling of speech signal nonstationarity. A hidden Markov model (HMM) is a stochastic process generated by two interrelated mechanisms, a Markov chain having a finite number of states, and a set of random functions which are associated with every transition between the states.

The states are connected by transitions. Each transition carries two sets of probabilities: a transition probability giving the probability of going from one state to another, and an output probability density function (pdf) which provides the probability of having an output symbol emitted when a transition is taken.

The observed sequence is assumed to be a stochastic function of the state sequence of the Markov chain. The state sequence itself is unobservable (hidden). The goal is to choose the parameters of the hidden Markov model to optimally match the observed characteristics of a given signal.

We could consider an HMM as a model for the vocal tract. For example, the articulatory positions of a vocal tract can be represented by the states of an HMM, and the changes in the vocal tract position can correspond to the transitions between the states of a model. When a transition is taken, a short signal is produced. This signal can have a finite number of possible characteristics which depend on the transition itself.

5.1 Discrete HMM vs Continuous HMM

The HMM models can be separated into two types: discrete density HMMs and continuous density HMMs.

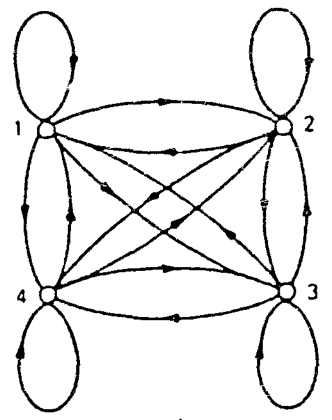
In the case of discrete density HMMs, the models are characterized by discrete pdf's. In order to use the discrete pdf's, each input frame must be represented by a symbol chosen from a finite alphabet. Therefore, vector quantization is used as a pre-processing stage for the discrete HMMs.

A second type of system is based on the continuous nature of speech samples. Usually, the speech parameters are in the form of multi-dimensional real-valued feature vectors (not vector quantized). By assuming certain properties for the vectors distribution, it is possible to estimate the output parameters from training data. Multivariate Gaussian density [24] is often used to calculate the probability density functions of the models. Many other forms of continuous densities can be applied [11] [31] [33].

The principal advantage of using continuous HMMs is the ability they have to directly model speech parameters. However, continuous HMMs require considerably longer training and recognition time. In this thesis, discrete HMMs are implemented. Although they are less flexible than continuous HMMs, and cannot recover from vector quantization errors, they are efficient and require less computations. Moreover, some useful information from vector quantization will be added to the HMM probabilities during the process of recognition.

5.2 HMM for isolated-word recognition

The most natural unit of speech is the word. Consequently, whenever it is possible, speech recognition systems are based on word models of speech. As it was explained in Section 2.1, word models can be used if the size of the vocabulary is small enough. In our case, the vocabulary is very limited, therefore, a distinct HMM is used for each word.



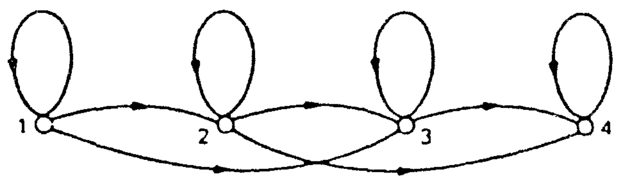


Figure 5.1: Representation of (a) an unconstrained HMM and (b) a Bakis HMM.

5.2.1 Basics of HMM

The most general case for a HMM model is the unconstrained HMM (Fig. 5.1(a)). In this model, every state can be reached with a single step from every other state. The unconstrained HMM can model any sequence of states, but for some applications like speech recognition, constrained HMMs have been found to model better the observed properties of the signals. The most commonly used constrained HMM for isolated-word recognition is the Bakis model [3] shown in Fig. 5.1(b). The Bakis model is a left-to-right type of HMM which has the desirable property that it can readily model signals whose properties change over time, e. g. , speech. As a matter of fact, this model is a sequence of states where each state could correspond, in theory, to some phonetic event, and each event could be skipped.

The characterization of a model is done by determining:

1. {N} - the number of states in the model.

   There are no clear rules to decide how many states are necessary to model a word. One idea is to let the number of states correspond roughly to the number of sounds (phonemes) within the word. Tests have shown that 5 or 6 states were a good choice for isolated-digit recognition [29] [10]. The HMM word model proposed in this work is the 5-state Bakis model(Fig. 5.2).

2. {M} - the number of distinct observation symbols in the alphabet of the model.

   In our case, the size of the alphabet is the number of codevectors used to vector quantize the signal.

3. $\{a_{ij}\}$ - the set of state transitions.

   a_{ij} is the probability of taking a transition from state i to state j:

$$a_{ij} = P(S_{t+1} = j, S_t = i) \tag{5.1}$$

where S_t is the state of the Markov chain at time t. The probability a_{ij} must also obey to the following conditions:

$$a_{ij} \geq 0 \qquad \forall i, j \tag{5.2}$$

$$\sum_j a_{ij} = 1 \qquad \forall i \tag{5.3}$$

Before the training of matrix A (a_{ij}), initial values for a_{ij} must be set. A simple uniform distribution is sufficient for initialization (assuming that the amount of training data is reasonable).

For uniform initialization, all transitions from a state are considered equally likely to be taken. Our matrix A, based on the transition probabilities of the model of Fig. 5.2, is initially set with the matrix shown in Fig. 5.3. The sum of the elements of each row must be equal to 1 according to Equation 5.3.

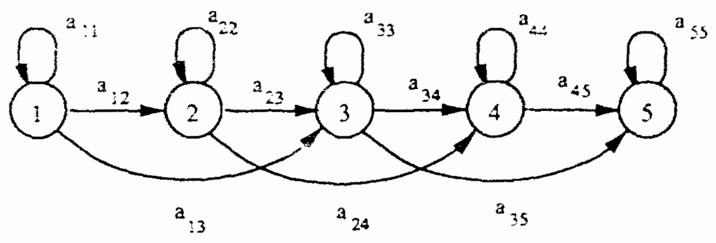


Figure 5.2: Representation of the 5-state Bakis model.

$$\begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 5.3: Uniform initialization of matrix A.

4. $\{b_{ij}(k)\}$ - the output symbol probability matrix.

b_{ij}(k) is the probability of emitting symbol k (one of the VQ indexes) when a transition is taken from state i to state j:

$$b_{ij}(k) = P(O_t = k | S_t = i, S_{t+1} = j) \tag{5.4}$$

O is the output sequence which is directly observed, and S is the state sequence which is *hidden*. So, O_t is the output symbol at time t and S_t is the state at time t.

The same probability conditions as for a_{ij}'s are applied:

$$b_{ij}(k) \geq 0 \qquad \forall i, j, k \tag{5.5}$$

$$\sum_k b_{ij}(k) = 1 \qquad \forall i, j \tag{5.6}$$

The matrix B is also uniformly initialized. Since k varies from 1 to M, and every symbol k has the possibility to be observed, all the initial elements of the matrix B are initially set to probability $1/M$.

5.3 Training the HMMs

There is no known analytical method to solve the problem of training the parameters of the HMM models. In fact, given any set of finite observation sequence as training data, there is no optimal way of estimating the model parameters. Instead, iterative procedures or gradient techniques [15] must be used. In our case, we used an iterative procedure; the Baum-Welch algorithm [4] or forward-backward procedure.

5.3.1 Baum-Welch algorithm

An HMM model is defined by its state transition probabilities (a_{ij}) and its output symbol probabilities ($b_{ij}(k)$). In this work, a_{ij} and $b_{ij}(k)$ were uniformly initialized. Uniform distribution was used because more complex initialization algorithms are not necessary for discrete density pdf's [12]. So, once the initial parameters are given, the model is reestimated iteratively with the Baum-Welch algorithm [4].

The Baum-Welch algorithm reestimation procedure is based on the intuitive notion that a new estimate of a state transition probability can be obtained from the expected number of transitions from state i to state j, divided by the expected number of transitions out of state i. Similarly, the new output symbol probability for the kth symbol is the expected number of transitions from state i to state j when symbol k is produced, divided by the expected number of transitions from state i to state j. The term expected is used because these statistics are usually averaged over large amount of data, and because the actual state transitions and output events are hidden [25]. The complete description of the algorithm is given in [32] [12] and can be summarized with the following equations.

Let's define the forward variable $\alpha_i(t)$ as:

$$\alpha_i(t) = P(O_1 O_2 \cdots O_t, S_t = i|\lambda) \tag{5.7}$$

where λ represents the model's parameters (matrix A and B). $\alpha_i(t)$ is the probability of having the partial observation sequence, $O_1, O_2, \cdots, O_t$, (until time t) *and* being in state i at time t, given the model λ. Given the training data, the forward probability can be recursively computed on t with:

$$\alpha_i(t) = \sum_j \alpha_j(t-1)a_{ji}b_{ji}(O_t) \tag{5.8}$$

In a similar manner, the backward probability $\beta_i(t)$ is defined as:

$$\beta_i(t) = P(O_{t+1}, O_{t+2}, \cdots, O_T | S_t = i, \lambda) \tag{5.9}$$

where $\beta_i(t)$ represents the probability of the partial observation sequences from time $t+1$ to the end, being in state i at time t and given the model λ. The backward probability can also be calculated recursively on t with:

$$\beta_i(t) = \sum_j \beta_j(t+1)a_{ij}b_{ij}(O_{t+1}) \tag{5.10}$$

Finally, it is useful to define the variable $\gamma_{ij}(t)$ as the probability of taking a transition from state i to j at time t given the entire observation sequence:

$$\gamma_{ij}(t) = P(S_t = i, S_{t+1} = j | O) = \frac{\alpha_i(t-1)a_{ij}b_{ij}(O_t)\beta_j(t)}{\alpha_{S_F}(T)} \tag{5.11}$$

where S_F is the final state.

Using these definitions, the reestimated state transition probabilities and output symbol probabilities $\bar{a}_{ij}$ and $\bar{b}_{ij}(k)$ are calculated by integrating jointly the current $\alpha_i(t)$, $\beta_i(t)$, a_{ij} and $b_{ij}(k)$. The estimates are given by:

$$\begin{aligned} \bar{a}_{ij} &= \frac{\sum_{t=1}^{T} \gamma_{ij}(t)}{\sum_{t=1}^{T} \sum_k \gamma_{ik}(t)} \qquad (5.12) \\ &= \frac{\text{expected number of transitions from state i to j given O}}{\text{expected number of transitions from state i at any time}} \end{aligned}$$

$$\bar{b}_{ij}(k) = \frac{\sum_{t:O_t=k} \gamma_{ij}(t)}{\sum_{t=1}^{T} \gamma_{ij}(t)} \tag{5.13}$$

$$= \frac{\text{expected number of transitions from state i to j given k}}{\text{expected number of transitions from state i to j for any k}}$$

Once the reestimation is done, $\bar{a}_{ij}$ and $\bar{b}_{ij}(k)$ are used as the new a_{ij} and $b_{ij}(k)$ for the next iteration. Good recognition results were obtained with two reestimations of the HMM parameters in the training procedure.

Smoothing

In our application, there is an insufficient amount of training data to estimate appropriately the model parameters. The frequently occurring output symbols may be well trained, but other symbols may be unobserved, and have zero probability. As a consequence, any probability multiplication having one unobserved output symbol probability may give a zero result. This is a critical problem, especially for recognition. So, a smoothing of the output pdf is essential.

The floor method [15] solves the problem of zero probabilities and works well for reasonably well trained models. If we had more difficult models to train, we could have used the distance method [35] or the co-occurrence method [12]. In the floor method, all the zero probabilities are replaced by a very small value (0.0001). To better smooth the pdf's, a linear interpolation is done between the estimates of the trained parameters and the minimum fixed values:

$$sb_{ij}(k) = \tau b_{ij}(k) + (1-\tau)P_m(k) \tag{5.14}$$

where $sb_{ij}(k)$ is the smoothed probability, and $P_m(k)$ is the minimum output symbol probability. τ is experimentally calculated to optimize the performance. In our HMMs, the best results were obtained with τ equal to 0.998.

5.4 Recognition with HMMs

In our application, each HMM is trained with word tokens. For recognition, we need to implement a procedure to search which model is the most likely to have produced the input observation sequence.

5.4.1 Viterbi algorithm

The Viterbi algorithm [41] was used to compute the output HMM probabilities. The probability of an HMM is defined as the best score along a single path of the model. This path corresponds in the model to the state sequence that has the highest probability of being taken while generating the observed sequence. Given the state sequence $S_1 S_2 \cdots S_t = i$ ending is state i and the observation sequence at time t, the HMM probability is defined with:

$$v_i(t) = \max_{S_1, S_2, \cdots, S_{t-1}} P(S_1 S_2 \cdots S_t = i, O_1 O_2 \cdots O_t | \lambda) \tag{5.15}$$

For recognition, the path probability is calculated recursively from the state transition probabilities a_{ij} and the output symbol probabilities $b_{ij}(k)$:

$$v_i(t) = \begin{cases} 0 & t = 0 \wedge i \neq S_I \\ 1 & t = 0 \wedge i = S_I \\ \max_j (v_j(t-1) a_{ij} b_{ij}(O_t)) & t > 0 \end{cases}$$

S_I is the initial state and $v_i(t)$ is the path probability at state i and time t.

The complete procedure to find the best state sequence is described in [32].

5.5 Multiple observation sequences

The procedures that were explained until now were based on the assumption that, at each frame, a single observation index is input into the HMM. In our system, multiple observation sequences are computed in the vector quantization stage. Recalling from Section 4.2.2, separate VQ indexes were calculated for each input feature vectors (cepstrum, differenced cepstrum, and energy vector). Consequently, the Baum-Welch algorithm and the Viterbi algorithm must be generalized for the multiple observation sequences.

Assuming that the observation sequences are independent, the generalization of the HMM training and recognition procedures can be done by computing the symbol probability as the product of the independent $b_{ij}(k)$'s obtained from each sequence:

$$b_{ij}(k) = \prod_{x=1}^{X} b_{ij}^{(x)}(k) \tag{5.16}$$

where $b_{ij}^{(x)}(k)$ is the output symbol probability from sequence x.

The use of multiple sequences increases the storage space for the B matrix. A separate set of $b_{ij}(k)$ needs to be stored for each of the multiple observation sequences. Nevertheless, the addition of supplementary storage is compensated by the fact that the precision of the signal description is significantly improved (see Section 4.2.2).

CHAPTER 6

ISOLATED-DIGIT RECOGNITION WITH VECTOR QUANTIZATION AND HIDDEN MARKOV MODELS

The three preceding chapters described the main stages of our isolated-digit recognizers. Now, we need to see how the feature extraction stage, the vector quantization stage and the HMM stage can be linked to each other and integrated into a system. This chapter is an overview of the different systems that have been designed and tested for isolated-digit recognition.

The first system uses VQ only for recognition. Vector quantization can be used for recognition [22] but it does not take into account the temporal characteristics of the utterances. This is a major flaw for a speech recognition system and to correct this problem, the rest of the proposed recognizers combine VQ and HMMs.

The first version of the VQ-HMM recognizer uses a single codebook to perform vector quantization. Single codebook systems were tested with three different types of codebook: one codebook made of universal codevectors, one made of word-specific codevectors and one made of phoneme-specific codevectors. The idea behind these tests was to find which type of codevectors performs the best in a VQ-HMM system.

The second version of VQ-HMM recognizers are made of word-specific VQs and word HMMs. Three word-specific VQ-HMM systems are presented. These systems were tested with the intention to illustrate that VQ can do more than simply compress data for discrete HMM recognizers by providing information that may be combined with the HMM probabilities in the process of the recognition decision.

To sum up, we will show that discrete HMMs give good performance for isolated-digit recognition, especially if word-specific VQ sequences are used as observation sequences.

6.1 VQ recognizers

6.1.1 Isolated-word recognizer

The basic idea in the following system is to perform recognition with vector quantization only [36] [38]. The suggested recognition system is illustrated on Fig. 6.1. The vector quantization is done with word-specific codebooks, i. e. , each word codebook is trained with many tokens of a same word contained in the training set. Recognition is done by choosing the word in the vocabulary whose average quantization distortion (according to its particular codebook) is minimum.

Three features are extracted from the input signal. The three features are vector quantized separately by multiple codebooks (Section 4.2.2). At each frame, the features give separate quantization distortions. The quantization distortions for the energy and the differenced energy are calculated separately. In order to obtain a single distortion score for each word, the quantization distortions are combined with a weighted sum; the composite distance metric [7]:

$$d_{tot} = w_{en} \times d_{en} + w_{den} \times d_{den} + w_{cep} \times d_{cep} + w_{dcep} \times d_{dcep} \tag{6.1}$$

d_{en}, d_{den}, d_{cep}, and d_{dcep} correspond respectively to the energy distortion, the differenced energy distortion, the cepstrum distortion and the differenced cepstrum distortion. The weights w_{en}, w_{den}, w_{cep}, and w_{dcep}, are determined by doing tests to optimize the recognizer's performance.

The composite distance is accumulated at each frame until all the frames of the input word are consumed. A final distortion score is obtained for each word. The

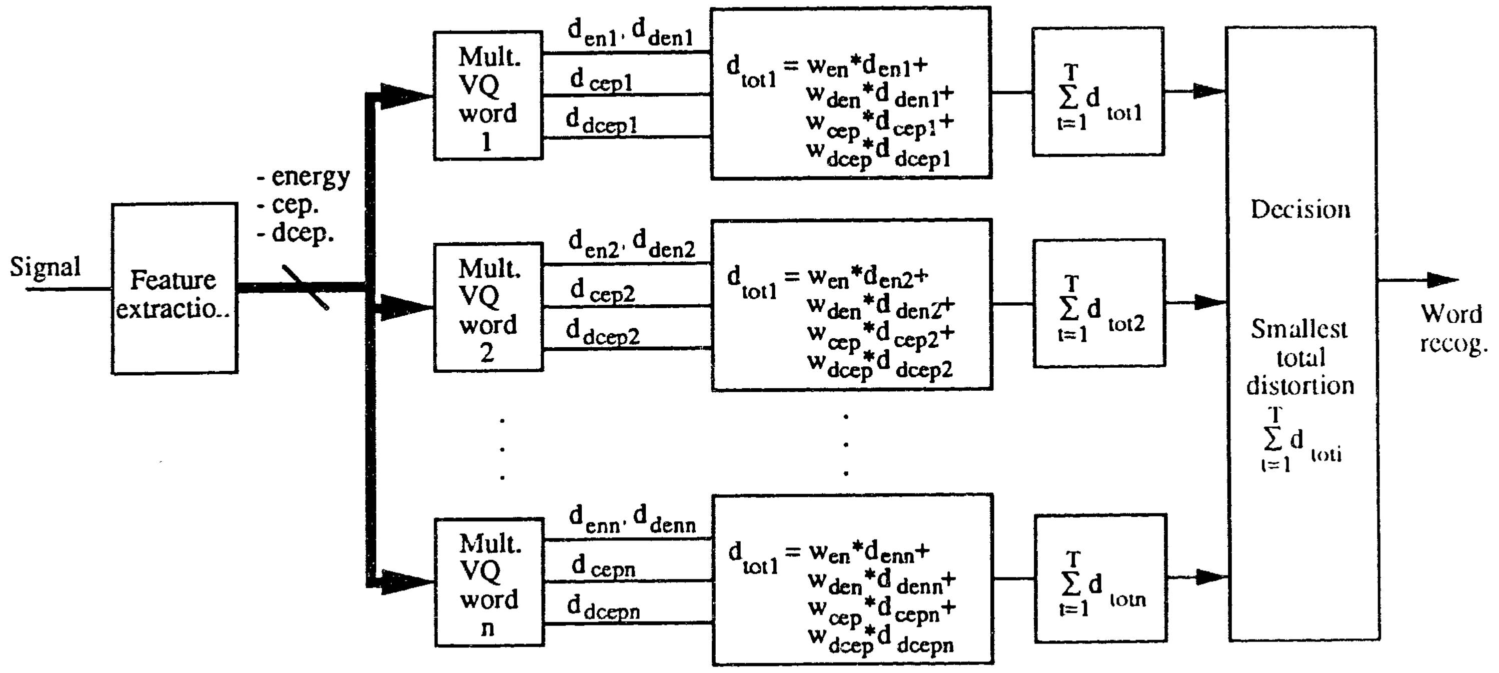

Signal
Feature extractio..
- energy
- cep.
- dcep.
Mult. VQ word 1
Mult. VQ word 2
Mult. VQ word n
d_{en1}, d_{den1}
d_{cep1}
d_{dcep1}
d_{en2}, d_{den2}
d_{cep2}
d_{dcep2}
d_{enn}, d_{denn}
d_{cepn}
d_{dcepn}
$d_{tot1} = w_{en} * d_{en1} + w_{den} * d_{den1} + w_{cep} * d_{cep1} + w_{dcep} * d_{dcep1}$
$d_{tot1} = w_{en} * d_{en2} + w_{den} * d_{den2} + w_{cep} * d_{cep2} + w_{dcep} * d_{dcep2}$
$d_{tot1} = w_{en} * d_{enn} + w_{den} * d_{denn} + w_{cep} * d_{cepn} + w_{dcep} * d_{dcepn}$
$\sum_{t=1}^{T} d_{tot1}$
$\sum_{t=1}^{T} d_{tot2}$
$\sum_{t=1}^{T} d_{totn}$
Decision
Smallest total distortion
$\sum_{t=1}^{T} d_{toti}$
Word recog.

Figure 6.1: Representation of a word-specific VQ recognizer.

smallest accumulated distance is used for recognition decision.

The word-specific VQ recognizer gives acceptable recognition rates for our small vocabulary databases. However, this recognizer fails to use temporal informations contained in speech signals.

6.1.2 Isolated-phoneme recognizer

The recognizer of Fig. 6.1 can be modified to obtain an isolated-phoneme recognizer by replacing the word-specific vector quantizers with phoneme-specific vector quantizers. Isolated-phoneme recognition can point out which phonemes are particularly difficult to recognize.

6.2 Single codebook VQ-HMM recognizer

As mentioned before, the major flaw of a VQ processor is the lack of capacity to use the temporal information of a speech signal. In the next system, we propose to introduce HMM models to correct this weakness of VQ. Vector quantization is used here as a pre-processing stage, for data reduction.

This section focuses on the implementation of three different single codebook vector quantizers for discrete HMM recognizers. A schematic diagram of a single codebook VQ-HMM recognizer is given on Fig. 6.2. An input feature vector is vector quantized by only one codebook and thus generates a single index that is used by each of the word HMMs. Only one codebook (single codebook) covers the whole vector space for a feature. In this section, three different single codebooks are proposed:

1. a universal codebook.
2. a global codebook made up of word-specific codevectors.
3. a global codebook made up of phoneme-specific codevectors.

A universal codebook is a single codebook generated from a training set including all the words of the vocabulary. The codevectors of a universal codebook are trained with feature vectors issued from each word. Universal codebooks are particularly suitable for large vocabulary applications since the whole feature vector space can be covered with one codebook.

In this thesis, single global codebooks are proposed as an alternative to universal codebooks, the idea being to improve the performance by having each codebook to cover more specific regions of the vector space. Two global codebooks are suggested, one obtained from the grouping of word-specific codebooks and the other from phoneme-specific codebooks. Here, grouping means that the codevectors of all the specific codebooks are put together into a single global codebook. The codevectors of such global codebooks are related more closely to the specific characteristics of the words or the phonemes.

A global codebook is schematized in Fig. 6.3. There are many codebooks but only one index (the one with the smallest quantization distortion) is extracted at every frame. In order to make a fair performance comparison, the total amount of codevectors (nx) is equal to the amount of codevectors in a universal codebook.

For the HMM stage, one model is generated for each word. The HMMs are

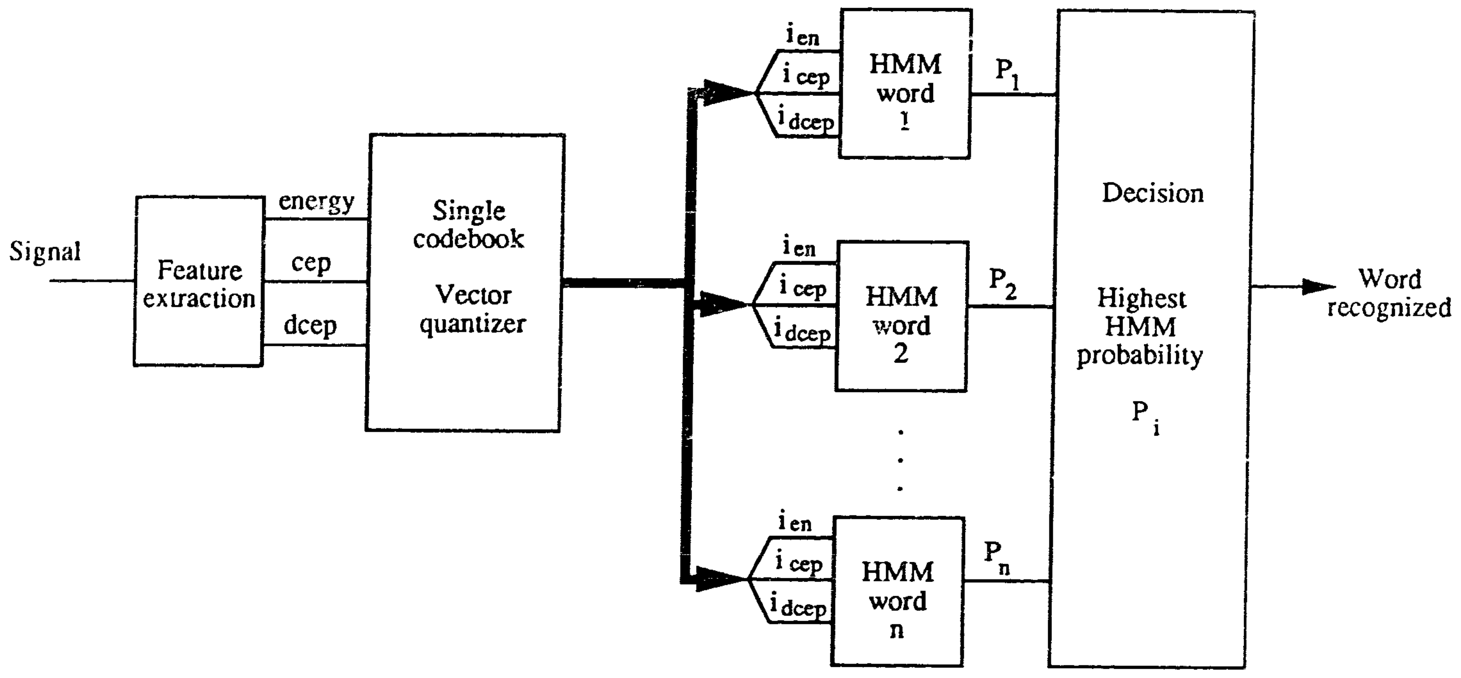


Figure 6.2: Representation of a single codebook VQ-HMM recognizer.

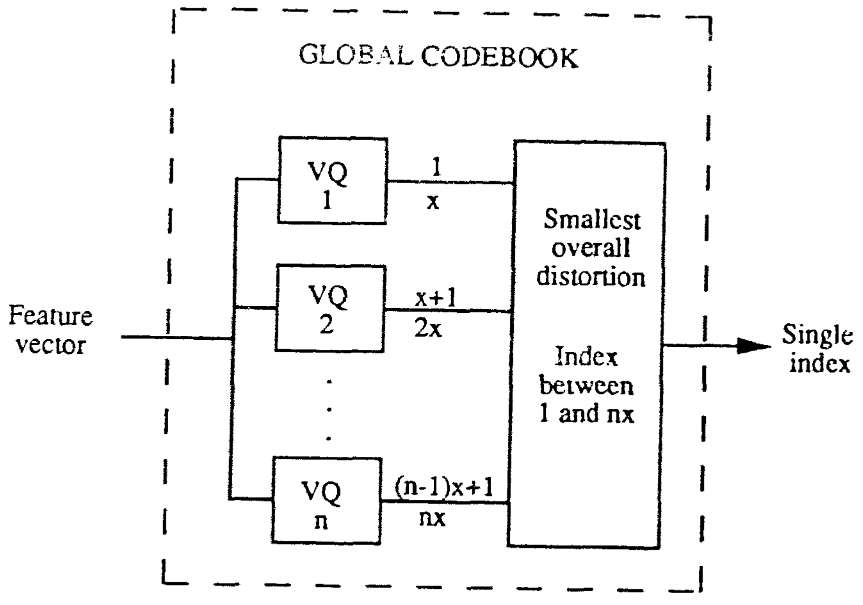


Figure 6.3: Representation of a global codebook.

trained with the Baum-Welch algorithm (Section 5.3.1) and recognition is done using the Viterbi algorithm(Section 5.4.1). The observation sequences used to train the word models are obtained from multiple codebooks vector quantizers (Section 4.2.2) and are assumed to be independent to each other. Therefore, the output symbol probabilities are multiplied together in the training and testing procedures (Section 5.5).

6.3 Word-specific codebook VQ-HMM recognizer

Many of the conventional discrete hidden Markov models (HMM) recognition systems use vector quantization (VQ) in the pre-processing stage. As mentioned above, the vector quantizer is usually composed of a single codebook, a universal codebook. We discuss in this section, how a single universal codebook can be replaced by several word-specific codebooks.

The motivation to make this change is based on the fact that VQs can do more than simply quantize the input signal. In addition to the VQ indexes, vector quantization can output distortion scores for each word. As a matter of fact, VQ distortion scores were successfully used to make isolated-word recognition with the system proposed in Section 6.1.1 (only for small vocabulary application). For that system, input words were processed through word-specific codebooks.

The input signal is quantized separately by each of the word-specific codebooks. Therefore, the HMMs can use word-specific observation sequences from each of the codebooks. Two interesting particularities emerge from such a system:

- The VQ and HMM stage can be combined consecutively as two discriminating techniques for recognition.
- The presence of more observation sequences can give a more detailed description of the input signal.

In this section, we exploit these two observations to test different ways of integrating the input feature coefficients into the framework of a discrete HMM recognizer. As is illustrated in Fig. 6.4, the problem consists of finding how word-specific index sequences and distortion measures can be used to make recognition with word HMMs.

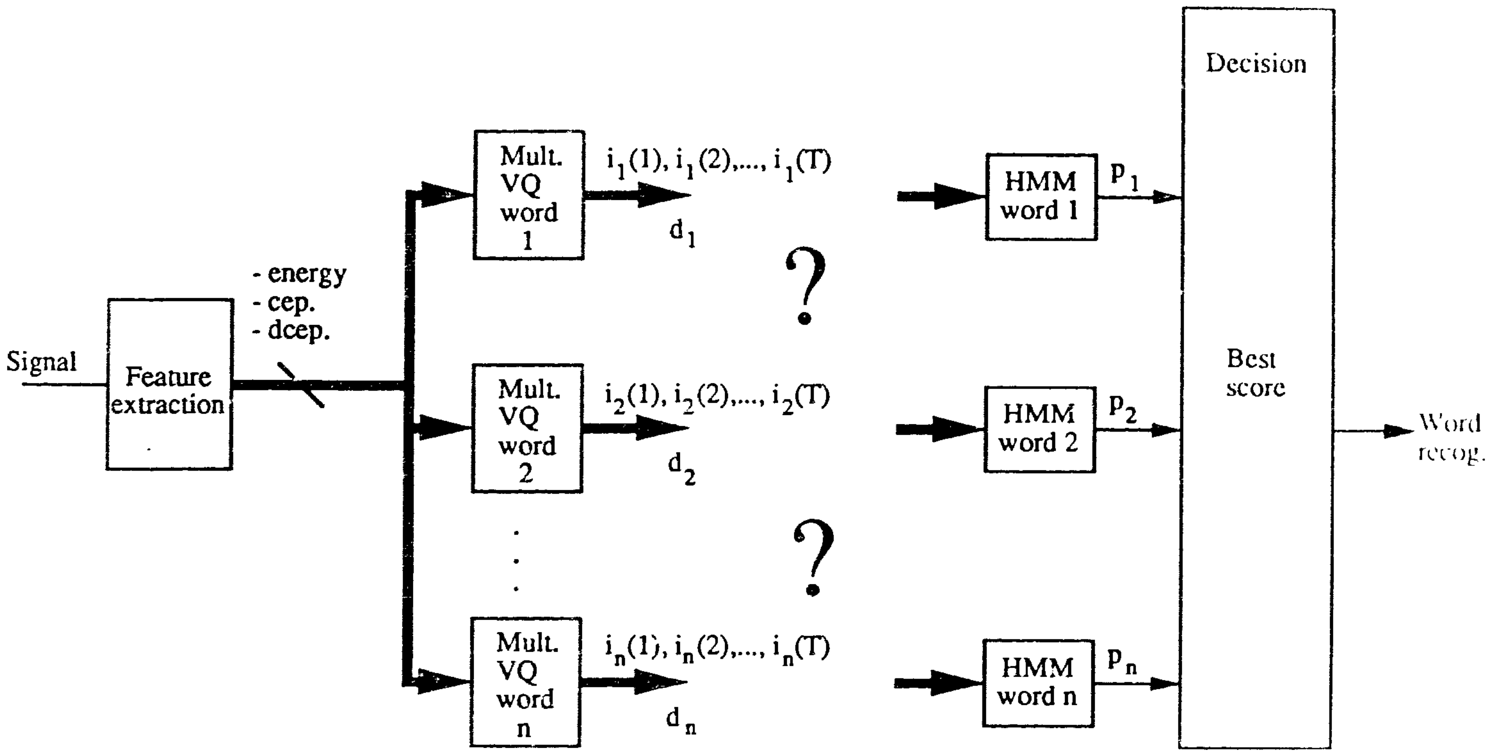


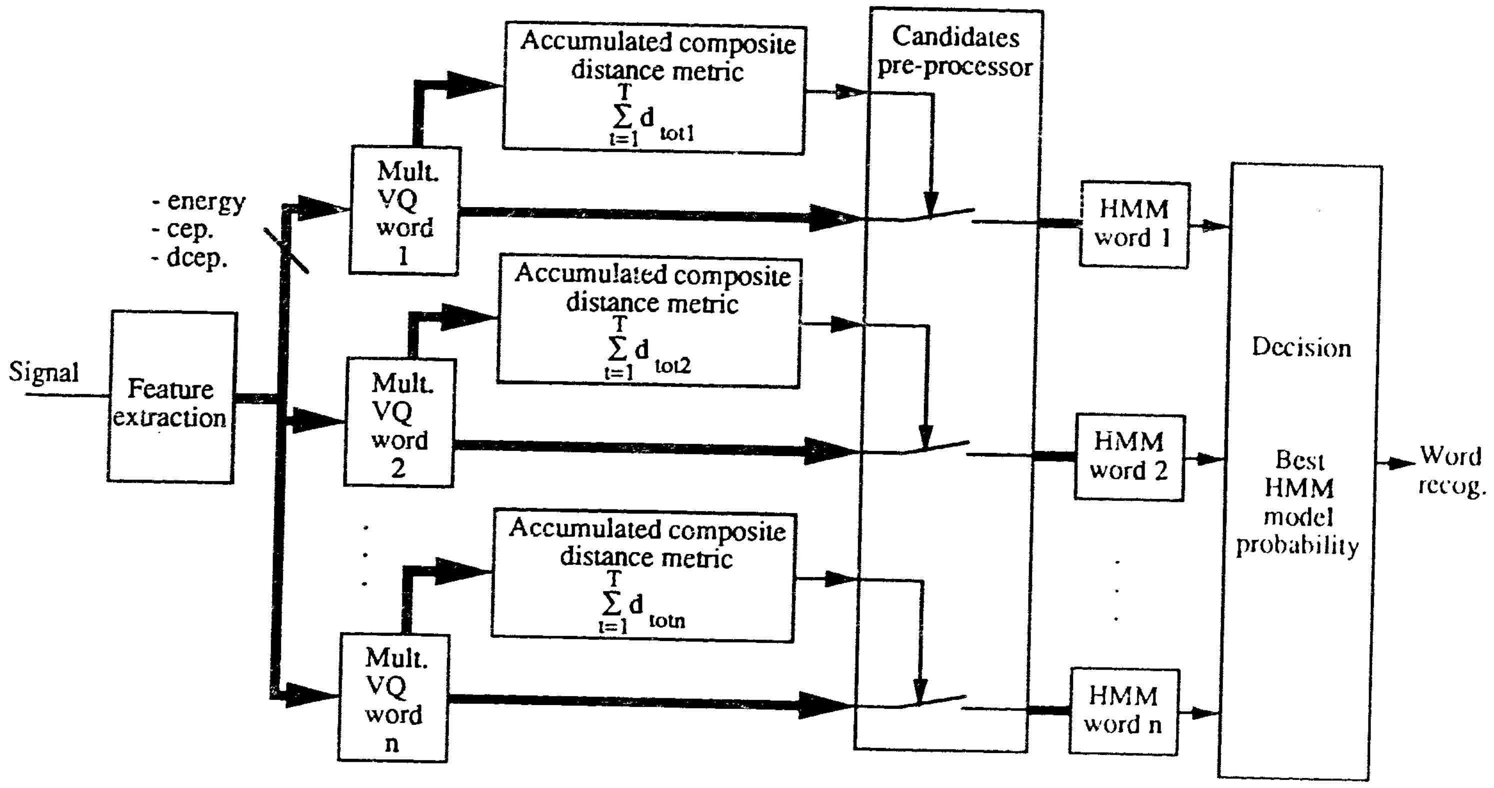
Figure 6.4: Representation of the word-specific VQ-HMM implementation problem.

6.3.1 Candidate selector VQ-HMM recognizer

The "candidate selector VQ-HMM recognizer" is depicted in Fig. 6.5. This is a recognizer where a processor is added to select the less distorted VQ candidates input to the HMM stage. The design of this recognizer is based on a parallel branch configuration; each word-specific VQ is connected to the corresponding word HMM by a direct and independent branch. Each branch represents the transmission of a word VQ multiple index sequence consisting of the cepstrum, the differenced cepstrum and the energy vector index sequences. A pre-processor is inserted on each branch to determine if a VQ sequence will be transmitted or not to the word HMM. The pre-processor gives more flexibility to the system by permitting the elimination of unlikely candidates from the VQ stage.

Each VQ sequence has an average quantization error score. Hence, the system can consider, for the transmission to the HMM stage, only the sequences with the lowest quantization error. As a consequence, the candidate selector allows a reduction in the number of calculations done in the HMM stage. Also, the recognizer is not confused by VQ sequences with high quantization error.

Tests have been made to find the optimal number of candidates that should be considered by the system.



Signal
Feature extraction
- energy
- cep.
- dcep.
Mult. VQ word 1
Mult. VQ word 2
Mult. VQ word n
Accumulated composite distance metric $\sum_{t=1}^{T} d_{tot1}$
Accumulated composite distance metric $\sum_{t=1}^{T} d_{tot2}$
Accumulated composite distance metric $\sum_{t=1}^{T} d_{totn}$
Candidates pre-processor
HMM word 1
HMM word 2
HMM word n
Decision
Best HMM model probability
Word recog.

Figu.e 6.5: Representation of the candidate selector VQ-HMM recognizer.

6.3.2 Combined VQ-HMM recognizer

In the introduction of Section 6.3, we noted that VQ and HMM information can be combined to improve recognition. The following system is used in order to verify this statement.

The word-specific VQ distortions are not used any more as candidate selectors but as word scores for the recognition decision. As illustrated on Fig. 6.6, a VQ distortion score and an HMM output probability are calculated for each word. A simple empirical formula is proposed in which the VQ and HMM scores are added together in a weighted sum. The probabilities are represented at logarithmic scale to reduce the dynamic range. The combination of VQ distortions and HMM probabilities is done with the following formula [5]:

$$R_i = C \times D_i + K \log(P_i) \tag{6.2}$$

where C is a positive constant, D_i is the distortion score for the codebook of word i, P_i is the output probability for the HMM of word i, and K is a negative constant computed to optimize the performance.

The formula allows recognition to be based on the VQ distortion scores only, HMM output probabilities only or both scores at the same time. Tests have been made with different values for C and K in order to optimize the performance. If the best performance is obtained using a combination of VQ and HMM information, we can conclude that VQ distortion scores provide recognition information that is complementary to the HMM probabilities.

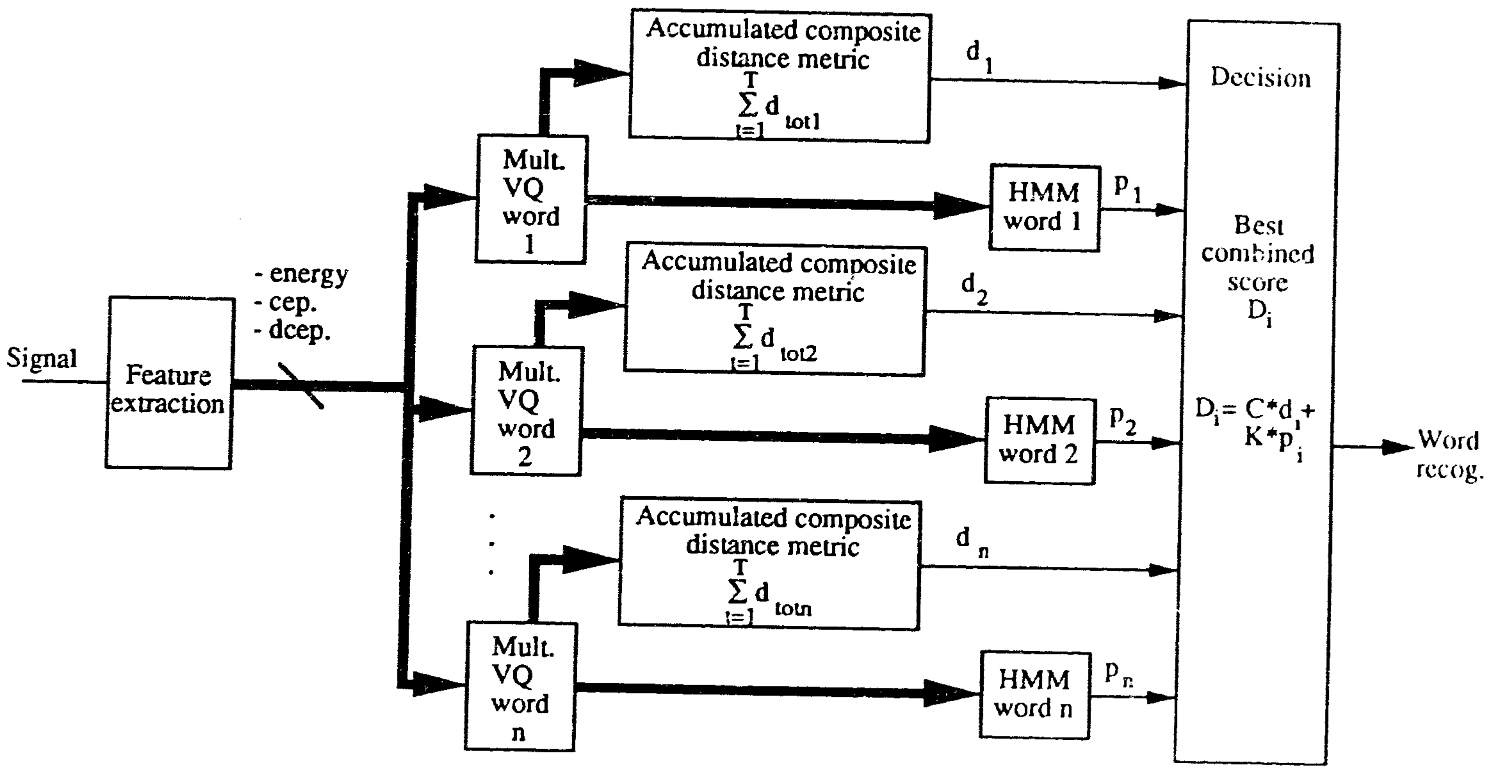

Signal
Feature extraction
- energy
- cep.
- dcep.
Mult. VQ word 1
Mult. VQ word 2
Mult. VQ word n
Accumulated composite distance metric $\sum_{t=1}^{T} d_{tot1}$
Accumulated composite distance metric $\sum_{t=1}^{T} d_{tot2}$
Accumulated composite distance metric $\sum_{t=1}^{T} d_{totn}$
HMM word 1
HMM word 2
HMM word n
d_1
d_2
d_n
p_1
p_2
p_n
Decision
Best combined score D_i
$D_i = C*d_i + K*p_i$
Word recog.

Figure 6.6: Representation of the combined VQ-HMM recognizer.

6.3.3 Fully interconnected word-specific VQ-HMM system

A second observation in the introduction of Section 6.3 states that more observation sequences can give a more detailed description of the input signal. The next system is proposed with the intention to make full use of the whole set of word-specific VQ sequences. The suggested system is called a "fully interconnected word-specific VQ-HMM recognizer" and is illustrated on Fig. 6.7.

In the fully interconnected VQ-HMM recognizer, the observation sequences applied to each HMM are not just the VQ outputs of a single word codebook, like in the case of the parallel word processing recognizer, but rather the outputs of all word-specific codebooks [40]. The increased amount of HMM input data makes the recognizer less dependent on the possible peculiarities of single observation sequences. This way, the recognizer is less vulnerable to the effects of a poor index sequence than in the case of the parallel word VQ-HMM recognizer. Also, the full interconnection may help to reduce the effect of the rough quantization at the VQ stage.

As usual, training is done with the Baum-Welch algorithm and recognition is done with the Viterbi algorithm. In both cases, the probability calculations must be modified to accept an increased amount of observation sequences simultaneously arriving into each word HMM model. The output probabilities are not calculated from only the multiple sequences of a single word VQ, but from the multiple sequences of all the word-specific VQs. Therefore, the output symbol probabilities of all the word-specific observation sequences must be combined to make a single Viterbi search for each word HMM.

Assuming that the word-specific VQ sequences are independent, we multiply the output symbol probabilities of all the word-specific VQ sequences using the relation:

$$p_i(t) = \max_j [p_j(t-1) a_{ji} \prod_w \prod_c b_{ji}(y_t^{wc})] \tag{6.3}$$

where y^{wc} is the observation sequence from word VQ w using the word codebook type c (energy, cepstrum or differenced cepstrum).

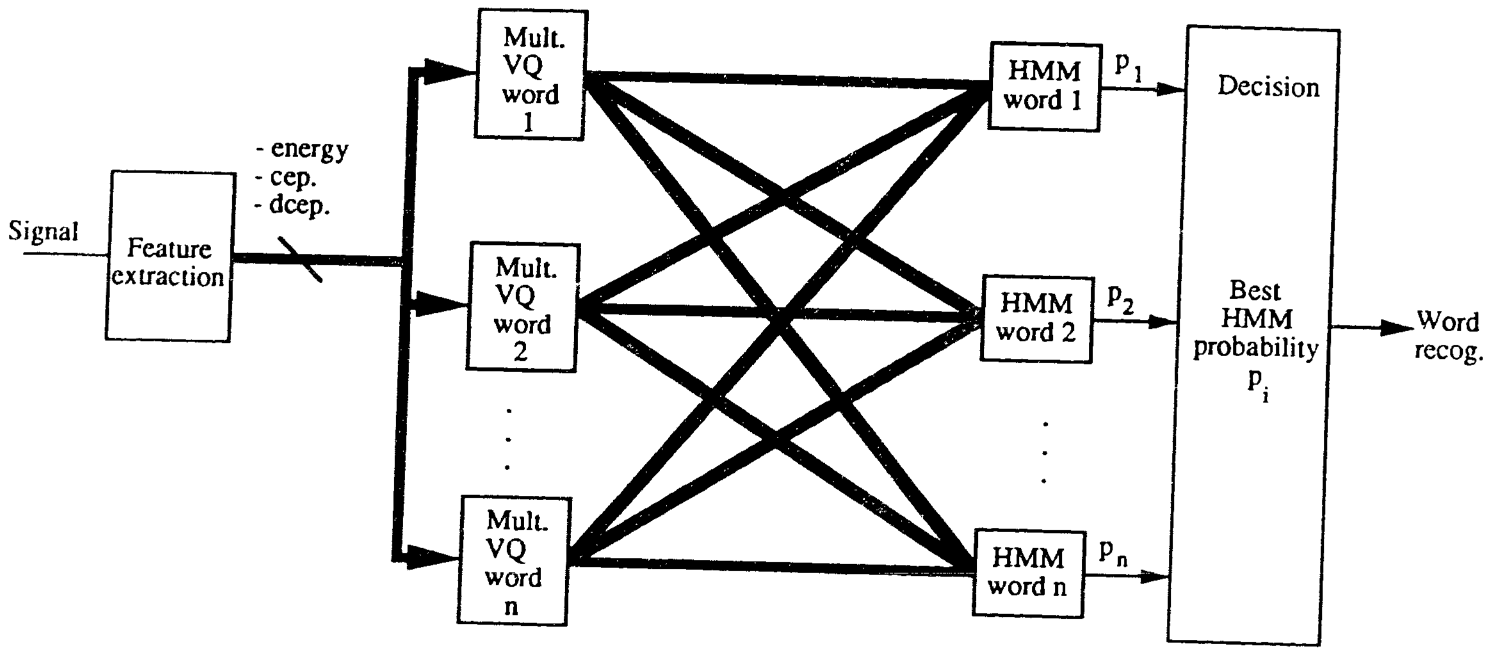


Figure 6.7: Representation of the fully interconnected VQ-HMM recognizer.

CHAPTER 7

EXPERIMENTAL RESULTS

7.1 Speech databases

Two speech databases were used to evaluate the different systems proposed in this thesis. One database was recorded in studio, and the other database was recorded over telephone lines. The next section describes the databases and specifies how the tokens of the databases were split into a training set and a testing set.

7.1.1 Studio database

The studio database, generated by Craig Scratchley [36], contains digits from zero to nine. Each digit is replicated two times by each of 20 talkers (10 male, 10 female). Each of the talkers is considered to have a fair 'Canadian' accent. Pauses were imposed between each words in the recording session. In other words, the tokens are isolated from any context dependency caused by neighboring words.

The recordings were done in SFU's recording studio in the Instructional Media Center. A special attention was put on the recording level, the length of pause between the words, the positioning of the microphone and the noise being picked up.

The digitization was done on a Sun-3 equipped with an ICS-100 digital signal processing board (12 bits quantization). First, the recorded words were passed through an analog anti-alias filter with a cutoff at around 8 KHz. The speech was sampled at 32 KHz and filtered with a digital filter with a 3.4 KHz cutoff. The digital signal was decimated by four, taking one sample at every four, to bring the rate at 8KHz. The resulting word tokens were stored in separate files.

Training set

The training set contains 10 talkers (5 male and 5 female) of the 20 talkers. Each digit is replicated two times by the same talker. For recognition, the tests are carried out with the tokens of the 10 remaining talkers.

7.1.2 Telephone database

The telephone database contains digits zero to nine with, in addition, the utterance "oh". All the tokens of the database are assumed to be spoken by a different talker. Each word is uttered by 50 talkers. Approximately 27% of the tokens are uttered by males, 33% by females and 40% by children. A subjective evaluation of the accents assumes that 81% of the talkers have a 'Canadian' accent and 19% have a foreign accent.

The tokens were obtained from customers calling to B. C. Tel yellow pages and asking for a service number of four digits. The telephone calls were recorded, and the resulting strings of digits were digitized and segmented into individual words. As a consequence, the beginning and the end of the words may be contextually dependent on the neighboring words.

Finally, phonemes were segmented from each word by keeping only the context independent frames. This means that only the frames considered to be independent from the neighboring phonemes were extracted. The transitional segments between two phonemes were rejected. The list of phonemes contained in the database is given in Table 7.1(a). The phonetic transcriptions of each digit is listed in Table 7.1(b).

The digitization and the segmentation of the utterances into words was done by MPR TelTech. The speech was sampled at 10KHz and quantized with 16 bits. An anti-alias filter was used with a cutoff frequency at 4.8 KHz. The resulting files were stored on disk, each file containing one word. All the details of the speech data processing are resumed in Appendix A.

Training set

The training set is made, for each digit, of 25 tokens randomly chosen. No consideration was given to the nature (sex, accent) of the talker. The tests were carried out with the remaining 25 tokens for each digit.

Phon.	*Example*	*Phon.*	*Example*
z	zoo	ao	bought
iy	beat	ay	bite
r	red	v	very
ow	boat	s	sis
w	wet	ih	bit
ah	butt	k	kick
n	non	eh	bet
t	tot	ax	the
uw	boot	ey	bait
th	thief	td	set
f	fief		

(a)

Word	*Phoneme sequence*
one	w-ah-n
two	t-uw
three	th-r-iy
four	f-ao-r
five	f-ay-v
six	s-ih-k-s
seven	s-eh-v-ax-n
eight	ey-td
nine	n-ay-n
zero	z-iy-r-ow
oh	ow

(b)

Table 7.1: List of phonemes (a) and single digits phonetic transcriptions (b)

7.2 Vector quantization recognition

7.2.1 Isolated-word recognition

The first results presented for isolated-digit recognition are obtained with a system based only on vector quantization. Vector quantization has some weaknesses for recognition applications but it's performance can still be used as a comparison basis to analyze the improvements generated by HMM-based systems.

The word-specific VQ recognizer has been described in Section 6.1.1. To implement the system presented, we need to determine two unknown parameters:

1. The weights of the composite distance metric.

2. The size of the codebooks.

Several tests have been done to find the optimal set of weights for the composite distance metric (i. e. the weights that optimize the performance) and to find the word codebook size giving the best result. The experiments were first done with the studio database and the recognition rates are listed in Table 7.2. The results show that:

- differenced cepstrum coefficients are the most reliable features for the present recognition task.

- energy and differenced energy both give poor performance. These features do not perform well when tested separately but they can still improve the performance when combined to the cepstrum or differenced cepstrum coefficients.

- there are no important variations in performance for the different sizes of codebook. A small amount of codevectors is sufficient for each codebook because of the small variety of speakers and recording conditions in the studio database.

Now, we must find which size of codebook and which set of weights performs well for our HMM-based systems using word-specific codebooks. The codebook size chosen for the studio database is 16. Sixteen codevectors per word gives a fair covering of the word vector space without generating excessive amount of computations for the vector quantizers. This codebook size is not too low for the studio database since very good results were obtained with the same database and 8 codevectors per codebook [36].

For the composite distance metric, we need to determine which set of weights should be used for recognition. Since we want to use size 16 codebooks, the set of weights was selected according to the best performance obtained with size 16 codebooks in Table 7.2. The best weights are (1,2,0.1,0.1). However, the results obtained suggest that energy is practically useless for recognition. The energy codevectors are inefficient and confuse the recognizer. Therefore, the energy feature was discarded from the composite distance metric. As a result, the final weights for the composite distance metric in all the following systems tested with the studio database will be (1,2,0,0.1).

In the case of the telephone database, we also fixed the word codebook size to 16. A bigger size for codebooks may have improved the performance since a higher number of codevectors could have vector quantized better the varied nature of the telephone database tokens. However, we limited the size of our codebooks to 16 because a high number of codevectors in the VQ stage generates a large amount

Recognition rates (%)						
w_{cep}	w_{dcep}	w_{en}	w_{den}	*Codebook size 8*	*Codebook size 16*	*Codebook size 32*
1	0	0	0	84	85.5	82.5
0	1	0	0	95.5	94	96
0	0	1	0	14	20	16.5
0	0	0	1	20	25.5	22.5
1	1	0	0	93.5	94.5	95
1	2	0	0	95	95	95
1	3	0	0	95.5	95	95
1	4	0	0	95.5	-	-
0	0	0.01	0.005	-	45	43.5
0	0	0.01	0.01	45	49.5	48
0	0	0.01	0.02	47.5	42.5	45
0	0	0.01	0.03	43.5	-	-
0	0	0.01	0.05	41	-	-
1	2	0.01	0.02	95	-	-
1	2	0.1	0.2	96.5*	-	-
1	2	0.4	0.8	96	-	-
1	2	0.01	0.01	-	95	95
1	2	0.1	0.1	-	95.5*	95.5
1	2	0.5	0.5	-	-	96
1	2	1	1	-	95.5	96.5*

Table 7.2: Isolated-word recognition results with word-specific VQ recognizer tested with the studio database

of symbols for the HMM discrete pdf's and therefore generates a large amount of computations and storage space for the HMM stage. With word codebook size fixed to 16, we obtained a performance of 78.2% using the weights (1,2,0,0.5). This low performance can be explained by the following factors: the recording quality for the telephone database is relatively poor; the tokens are corrupted by different levels and types of background and transmission noise; the variety of speakers (male, female, children, foreign accents); and the large percentage of children speech result in a very difficult recognition task.

7.2.2 Isolated-phoneme recognition

Isolated-phoneme recognition has been performed with a system similar to the isolated-word recognizer of Fig. 6.1 but with phoneme-specific codebooks instead of word-specific codebooks. Since the variety of sounds within a phoneme is smaller than for words, the number of codevectors in each codebook can be smaller for phonemes than for words. In our case, the codebook size has been set to 8 codevectors.

First, each feature for each phoneme has been tested separately. The tests were done to check how the VQ recognizer performs when either one of the features is used. These tests correspond to the case when only one weight in the composite distance is different from 0.

Table 7.3 shows the recognition rate for each phoneme (rows) when either one of the features (columns) is used. The recognition rates are generally low as we can see from the average rate for each feature. In general, we can say that the cepstrum coefficients give the best results and the energy is useful for only a few phonemes.

Other tests were done for isolated-phoneme recognition, this time by combining together the different feature distortion scores of a phoneme. An arbitrary set of composite distance metric weights was fixed despite the fact that these weights may work better for some phonemes than others. The best results were obtained with the weights $\{w_{cep},w_{dcep},w_{en},w_{den}\}=$ (1,1,0.1,0.5). These weights are slightly different to the optimal weights (1,0.5,0,0.3) and (1,0.8,0.01,0.05) obtained from similar phonetic recognition experiments [12].

The results of the isolated-phoneme recognizer using the composite distance metric are given in the form of a confusion matrix. Table 7.4 contains the confusion matrix where the x axis shows the phonemes recognized when the phonemes of the y axis are tested. The recognition rates are given in percentage.

The results illustrated in the confusion matrix of Table 7.4 suggest the following remarks:

- The phonemes sounding similar, like "ao" and "ow", are very likely to be confused.
- The fricatives ("th", "f", "v", "s", "z") are very difficult to differentiate from each other.
- The phonemes "th" and "z" are particularly difficult to recognize.

The low results obtained for phoneme "th" and "z" may be caused by a bad choice of weights for the composite distance metric. Therefore, we tested the two phonemes for different sets of weights. In the case of phoneme "z", the maximum

individual phoneme recognition rates (%)				
Phoneme	*Features*			
	Cepstrum	*Diff. cep.*	*Energy*	*Diff. en.*
w	92	60	0	48
ah	32	40	0	0
n	77	47	15	24
t	48	28	0	12
uw	72	60	52	8
th	32	4	0	0
r	84	21	0	15
iy	30	32	8	10
f	64	28	0	22
ao	36	36	0	24
ay	36	65	10	12
v	38	2	42	20
s	37	32	11	5
ih	44	36	8	20
k	36	80	28	12
eh	20	20	0	52
ax	36	28	32	28
ey	40	8	0	0
td	56	36	40	20
z	4	4	0	4
ow	26	22	0	2
Average	44.8	32.8	11.7	17.1

Table 7.3: Isolated-phoneme recognition results with the telephone database

Phoneme tested (y axis) vs Phoneme recognized (x axis)																					
	w	**ah**	**n**	**t**	**uw**	**th**	**r**	**iy**	**f**	**ao**	**ay**	**v**	**s**	**ih**	**k**	**eh**	**ax**	**ey**	**td**	**z**	**ow**
w	96														4						
ah	4	52					12				32										
n			78	1	6			2				10	2							1	
t				44		16						4	16						4	16	
uw				4	96																
th			4	20		24			4			12	20							16	
r		4		4		3	72			1	1	3		1			5			5	
iy			4		8		8	40						24			2	14			
f				2		2			52		10	18		16							
ao										76			4								20
ay		6					4				82			2			6				
v			26				2					66								6	
s				9		1			17			10	56		7						
ih				4	20			8						56		8	4				
k				4					8						88						
eh					4		12			4	8			8		56	4	4			
ax							16			8		12					64				
ey			4		32		4	4						8				48			
td									4			4			40				52		
z			8						16			36	16							24	
ow		2			4		6			34		4					6				44

Table 7.4: Confusion matrix for the phonemes of the telephone database

recognition rate obtained remained at 24% (Table 7.5). For phoneme "th", an improvement is obtained when the weights are changed. The recognition rate for "th" raised from 24% to 40% (Table 7.6).

From the confusion matrix, it is possible to calculate an overall isolated-phoneme recognition performance by averaging all the phoneme recognition performance together (by taking the averages of the terms along the diagonal). The overall recognition rate is 60.6% which is satisfying for isolated-phoneme recognition if we compare that result to the performances of other phoneme recognizers [12].

The low recognition rates for isolated-phoneme recognition suggest that isolated-word recognition with a small vocabulary is easier to perform with word units than subword units. Effectively, the recognition rates for isolated-word recognition are higher than for isolated-phoneme recognition. Moreover, subword units introduce an additional burden by imposing the implementation of a procedure to reconstruct the words from phonemes.

Recognition rates (%)				
w_{cep}	w_{dcep}	w_{en}	w_{den}	*Results*
1	0	0	0	4
0	1	0	0	4
0	0	1	0	0
0	0	0	1	4
1	0.5	0	0	12
1	1	0	0	24*
1	1.5	0	0	20
1	2	0	0	20
1	1	0	0.2	24
1	1	0	0.5	24
1	1	0	1	20
1	1	0.5	0	24
1	1	1	0	24
1	1	3	0	20

Table 7.5: Isolated-phoneme recognition results for phoneme "z"

Recognition rates (%)				
w_{cep}	w_{dcep}	w_{en}	w_{den}	*Results*
1	0	0	0	32
0	1	0	0	4
0	0	1	0	0
0	0	0	1	0
1	0.1	0	0	32
1	0.5	0	0	32
1	0.6	0	0	32
1	0.8	0	0	28
1	1	0	0	24
1	0	0	0.5	32
1	0	0	1	36
1	0	0	2	28
1	0	0.3	1	36
1	0	0.5	1	40*
1	0	0.7	1	36
1	0	1	1	32
1	0.5	0.5	1	28

Table 7.6: Isolated-phoneme recognition results for phoneme "th"

7.3 Single codebook VQ-HMM recognition

In this thesis, three single codebook VQ-HMM recognizers are studied (Section 6.2). We want to find which type of codevectors, either from universal codebooks, word-specific codebooks or phoneme-specific codebooks, gives the best performance for a discrete HMM recognizer.

Based on the results of the VQ recognizers, we have set the size of the word-specific codebooks to 16 and the size of the phoneme-specific codebooks to 8. For universal codebooks, the size was set according to the total number of codevectors in the global codebooks.

For the studio database, only universal codebooks and word-specific codebooks have been trained since no phoneme tokens were available. The size of universal codebooks was fixed to the total number of codevectors contained in the word-specific codebooks(10 words $\times$ 16 codevectors/word = 160 codevectors).

For the telephone database, 11 word-specific codebooks and 22 phoneme-specific codebooks were trained. An additional phoneme codebook was trained with silent frames. The addition of a silence codebook is justifiable by the fact that many frames of the input tokens are silent. The total number of codevectors for the word-specific (11 $\times$ 16) and phoneme-specific (22 $\times$ 8) global codebooks is 176 codevectors. So, universal codebooks of size 176 were trained.

The results of the three single codebook VQ-HMM recognizers are shown in Table 7.7. Two observations can be made:

- The addition of HMMs improves the performance. When we compare the

Recognition system	*Studio*	*Telephone*
Universal	97.5%	83.6%
Global word	98.0%	81.8%
Global phoneme	-	77.8%

Table 7.7: Isolated-word recognition with single codebook VQ-HMM systems

performance of the best single codebook VQ-HMM system to the best results of the VQ recognizers, we obtain an improvement of 1.5% (from 96.5% to 98%) for the studio database and an improvement of 5.4% (from 78.2% to 83.6%) for the telephone database.

- The results are unclear to know which single codebook VQ gives the best performance. The studio database shows that the word-specific global codebook gives slightly better results than the universal codebook, whereas the telephone database gives opposite results; the word-specific and phoneme-specific global codebooks don't perform as well as the universal codebook recognizer. The contradiction in the results may be explained by the differences between the two databases.

The telephone database contains a larger variety of talkers than the studio database. Therefore, the feature vectors of the telephone database may be more sparsely spread in the vector space than those of the studio database. If the feature vectors are sparsely spread, they should be vector quantized with codevectors covering well the whole vector space. The codevectors of universal codebooks are usually well trained for such cases; that may explain the superiority in performance of universal codebooks for the telephone database.

On the other hand, word-specific and phoneme-specific codebooks are trained to cover restricted regions of the vector space. Each codebook is trained independently from the other codebooks. This means that different codebooks may cover intersecting regions of the vector space. So, codevectors of different codebooks can be positioned very near to each other. This lets some big areas of the vector space to be uncovered by the codebooks. If the input vectors fall into these critical regions, the quantization error will be high. If the input vectors stay within the regions covered by the codebooks, quantization is satisfying. The previous argument may explain why, for the studio database, the word-specific codebook VQ gave good results. Since the talkers of the studio database are not very diversified, the feature vectors may stay in specific regions of the vector space.

We intend, in the next section, to still use word-specific codebooks for vector quantization, but this time, the system conceived will exploit better the advantages that word-specific VQs can provide to isolated-word recognition.

7.4 Word-specific codebook VQ-HMM recognition

The next recognition systems have a common characteristic, they all use word-specific codebooks to generate a set of word-specific VQ sequences. In Chapter 6, three word-specific VQ-HMM recognizers were presented. The results are summarized in the following sections.

7.4.1 Recognition with the candidate selector VQ-HMM system

For the "candidate selector VQ-HMM system" (Section 6.3.1), a distortion score is calculated for each word. The distortion score reflects the quantization distortion associated to a word. Only the candidates giving the lowest quantization distortions are selected for transmission to the HMM stage. Tests were carried out for every possible number of candidate pre-selection. When only one candidate is selected, recognition is uniquely based on the VQ scores. When all the candidates are considered, the VQ scores are ignored.

The tests are made with word-specific codebooks of 16 codevectors. And, as it was mentioned in Section 7.2.1, we now use the composite distance metric weights (1,2,0,0.1) for the studio database and (1,2,0,0.5) for the telephone database. Experiments were conducted with the two databases and the results are shown in Table 7.8. The results suggest that:

- Generally, the performance decreases when the number of candidates increases. This means that the parallel branch configuration gives better results with VQ distortion scores than with HMM output probabilities.
- The best performance for the telephone database is obtained when 2 candidates are selected. This result suggests that a candidate selector pre-processor can be justified for some applications.

The overall performance of the candidate selector VQ-HMM system is bad because the parallel branch system fails to take advantage of the HMM stage; the best

Nb. of candidates	1	2	3	4	5	6	7	8	9	10	11
Studio	94	92	89.5	87	85.5	83	81	79.5	78.5	76	-
Telephone	78.2	79.3	76.7	75.3	75.6	74.9	74.5	73.8	73.8	73.4	73.1

Table 7.8: Recognition rates for candidate selector VQ-HMM system

performances were obtained when only one candidate is selected from the VQ stage and transferred to the HMM stage. In such a case, the recognition decision is done at the VQ stage only. The next suggested system, the combined VQ-HMM, makes a more direct use of the HMM and VQ scores.

7.4.2 Recognition with the combined VQ-HMM system

The parallel branch configuration is again tested and all the word VQ sequences are connected to the word HMMs. The VQ distortion scores and the HMM output probabilities are combined with the following equation (see Section 6.3.2):

$$R_i = C \times D_i + K \log(P_i) \qquad (7.1)$$

The testing conditions and the results are summarized in Table 7.9. When $C = 0$ and $K = 1$, only the HMM probabilities are used for the final decision. When $C = 1$ and $K = 0$, only the VQ distortions are used. The system was tested again with weights (1,2,0,0.1) for the studio database and (1,2,0,0.5) for the telephone database (see Section 7.2.1).

The results of Table 7.9 show that:

C	K	*Studio*	*Telephone*
1	0	94	78.2
1	-0.0000001	94.5	-
1	-0.000008	98.5*	82.9
1	-0.00001	98.5	84*
1	-0.00002	98	83.27
1	-0.00005	96.5	83.27
0	-1	76	73.1

Table 7.9: Recognition rates for the combined VQ-HMM system

- an improvement is achieved when D_i and P_i are combined together, rather than used separately. This means that VQ distortion scores contain useful complementary information to the HMM probabilities. A more sophisticated method, like neural networks, could combine even better these pieces of information than the simple equation proposed for this system.

- we obtained the best overall performances so far. Effectively, this system is superior by 0.5% for the studio database and 0.4% for the telephone database to the best single codebook VQ-HMM system (see Table 7.7). There is an improvement despite the fact that the parallel configuration gives low performance (76% or 73.1%) when only HMM probabilities are used.

C	K	*Studio*	*Telephone*
1	-0.0000001	97.5	-
1	-0.000005	-	81.1
1	-0.00001	97.5	-
1	-0.00005	98	81.4
1	-0.0001	98*	81.8*
1	-0.0005	-	81.8
1	-0.001	-	81.8
0	-1	98	81.8

Table 7.10: Recognition rates for the global codebook and VQ-HMM combination

7.4.3 Recognition with word-specific global codebook and VQ-HMM information combination

The word-specific global codebook system (Section 6.2) can generate word-specific VQ distortion scores. We know from previous results that the global codebook system achieves good performance, so, it is interesting to verify if combining VQ distortions and HMM probabilities improves the performance of that system. The VQ and HMM scores are combined again using Equation 7.1.

The results of Table 7.10 indicate that no improvement is obtained from the combination of VQ and HMM informations for the global word-specific codebook system (when compared to Table 7.7).

Nb. of candidates	1	2	3	4	5	6	7	8	9	10	11
Studio (K=-8*10^{-6})	94	97.5	98	98.5	98.5	98.5	98.5	98.5	98.5	98.5	-
Telephone (K=-1*10^{-5})	78.2	82.2	83.6	83.3	84	84	84	84	84	84	84

Table 7.11: Recognition rates with candidate selector and VQ-HMM information combination

7.4.4 Recognition with candidate pre-selection and VQ-HMM information combination

The idea for the next system is to join together in a parallel branch system, a candidate selector and a combination of VQ distortion scores with HMM probabilities. The system is identical to the recognizer shown in Fig. 6.5, but in addition, the accumulated composite distance metrics are transferred to the final decision block. This means that the VQ distortion scores are used for candidate selection and for the final recognition decision. The tests were done with the optimal combination factor K obtained in Table 7.9.

We denote from this system no amelioration of performance, i. e. , the "all candidates" recognizer gives the optimal rates (see Table 7.11). Nevertheless, we can observe that a limited number of candidates show also optimal performances. As a matter of fact, 5 candidates prove to be sufficient. Therefore, with the candidate selector, the computational complexity can be cut by half without affecting the final results.

7.4.5 Recognition with the fully interconnected word-specific VQ-HMM system

The last recognition experiment was carried out with the fully interconnected word-specific VQ-HMM system discussed in Section 6.3.3. The recognizer is built to make full use of the word-specific VQ sequences. As a matter of fact, each HMM model is trained and tested with the whole set of word-specific VQ index sequences.

The results for this system are: 99.5% recognition rate for the studio database and 85.8% for the telephone database. In both cases, the performance is superior to the best performance obtained so far. In fact, the current results are the best results that are achieved in this thesis. The results imply that:

- a higher number of observation sequences can significantly improve the performance (about 2% of improvement when compared to the best result of the single codebook VQ-HMM system).
- a word HMM is better trained by observing the whole set of word-specific VQ sequences than by observing a single observation sequence.

The advantages of this system are obtained at the expense of a higher amount of computations, since more index sequences need to be processed in the HMM stage. When we compare the computational complexity of the fully interconnected configuration to the parallel configuration, we realize that although the VQ stage requires the same amount of computations in both cases, the fully interconnected configuration requires approximately ten times more computations than the parallel configuration for the HMM stage. The amount of additional computations is proportional to the size of the vocabulary. Most of the additional computations come from the multiplication of the independent output symbol probabilities (see Section 6.3.3). The number of computations is acceptable as long as the vocabulary is small. which is the case for our databases.

CHAPTER 8

CONCLUSION

The objective of this thesis was to investigate a reliable isolated-digit recognizer. Relying upon the state of the art research, we decided to utilize vector quantization and hidden Markov models. The recognition procedure was divided into three stages: feature extraction; VQ; and HMM. The basic components of each stage were first described and we then proposed systems linking the above three stages.

In this thesis, additional VQ information was combined with HMM information to perform recognition. The use of distortion scores was proposed as an alternative to conventional VQ-HMM systems where VQ is simply used as a pre-processing stage for HMMs. We suggested three types of recognizer: VQ recognizers, single codebook VQ-HMM recognizers and word-specific codebook VQ-HMM recognizers.

VQ recognizers were tested to check the usefulness of VQ distortion scores for recognition. Then, single codebook VQ-HMM recognizers were introduced to show how HMM models can improve the performances of a VQ recognizer. Finally, word-specific codebook VQ-HMM recognizers were proposed with the intention to verify if VQ distortion scores can be added in the process of the recognition decision when HMMs are used. The word-specific codebook VQ-HMM recognizers were also proposed to check if the performances of an HMM system could benefit from an increased number of observation sequences.

We found out that VQ distortion scores helped to improve the performance of our system. We concluded that VQ information is complementary to HMM information and can be advantageously used for recognition. The results also showed that an improvement is obtained when each HMM is trained with a set of word-specific VQ index sequences. These results suggest that multiple VQ sequences have advantages over a single VQ sequence systems. However, the implementation of more sequences must respect the limits imposed by the computational complexity and the processing time allowed.

The recognition rates obtained were very good when the recognizer was tested with a studio recorded database. The results went up to 99.5% recognition. When we used a database recorded over telephone lines, the results were 85.8% in the best case. This means that more work has to be done to overcome the difficulties present in noisy and irregular environments such as telephone lines.

Appendix A:

Telephone Database Speech File Pre-Processing

Once the telephone database tokens were digitized on 16 bits, they were converted to 12 bits by discarding the 4 least significant bits, in order to be compatible with our D/A converter.

The sampled data, in binary format, was stored on two-byte unsigned integers. The integers ranged from 0 to 32767 (0 to 7FFF Hex) with the low-byte coming first. Zero amplitude was represented by the value 16384 (4000 Hex). The low-byte and the high-byte were inverted. The 16 bits integers were finally divided by 8 to be stored on 12 bits (0 to 4095).

The word tokens and the phoneme tokens were segmented with MacSpeech Lab II (Macintosh). The resulting tokens were put in individual files and transferred back to the Sun system. The final digital signals were centered around 0 (signal range ± 2048) which made possible their audition with the D/A converter of the ICS-100 board.

REFERENCES

[1] B. S. Atal and M. R. Schroeder, "Predictive Coding of Speech Signals," *Proc. 6th Int. Cong. Acoust.*, C-5-4, 1968.

[2] J. K. Baker, "The dragon system-An overview," *IEEE Trans. on Acoust., Speech, Signal Processing*, ASSP-23(1):24-29, February 1975.

[3] R. Bakis, "Continuous Speech Recognition via Centisecond Acoustic States," *91st Meeting of the Acoustical Society of America*, April 1976.

[4] L. E. Baum, "An Inequality and Associated Maximization Technique in Statistical Estimation of Probabilistic Functions of Markov Processes," *Inequalities* 3:1-8, 1972.

[5] Y. S. Cheung and S. T. Leung, "Speaker Independent Isolated Word Recognition Using Word-Based Vector Quantization and Hidden Markov Models," *in Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, pp. 27.4.1-27.4.4, April 1987.

[6] L. Cossette, E. Velez and V. Cuperman, "Speaker-Independent Isolated-Digit Recognition based on Hidden Markov Models and Multiple Vocabulary Specific Vector Quantization," *in Proc. IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp. 146-149, May 1991.

[7] S. Furui, "Speaker-Independent Isolated Word Recognition Using Dynamic Features of Speech Spectrum," *IEEE Trans. on Acoust., Speech, Signal Processing*, ASSP-34(1):52-59, February 1986.

[8] S. Furui, *Digital Signal Processing, Synthesis, and Recognition*, New York, Marcel Dekker, Inc., 1989.

[9] V. N. Gupta, M. Lennig and P. Mermelstein, "Integration of Acoustic Information in a Large Vocabulary Word Recognizer," *in Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, pp. 697-700, April 1987.

[10] B. H. Juang, L. R. Rabiner, S. E. Levinson and M. M. Sondhi, "Recent Developments in the Application of Hidden Markov Models to Speaker-Independent Isolated Word Recognition," *IEEE Int. Conf. Acoust., Speech and Signal Processing*, pp. 1.3.1-1.3.4, April 1985.

[11] B. H. Juang and L. R. Rabiner, "Mixture Autoregressive Hidden Markov Models for Speech Signals," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-33(6):1404-1413, December 1985.

[12] K. F. Lee, *Automatic Speech Recognition, The Development of the SPHINX System*, Norwell, MA, Kluwer Academic Publisher, 1989.

[13] K. F. Lee and H. W. Hon, "Speaker-Independent Phone Recognition Using Hidden Markov Models," *IEEE Trans. on Acoust., Speech, Signal Processing*, ASSP-37(11):1641-1648, November 1989.

[14] K. F. Lee, H. W. Hon and R. Reddy, "An Overview of the SPHINX Speech Recognition System," *IEEE Trans. on Acoust., Speech, Signal Processing*, vol. ASSP-38, no. 1, pp. 35-45, January 1990.

[15] 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," *Bell Syst. Tech. J.*, vol. 62, no. 4, pp. 1035-1074, April 1983.

[16] Y. Linde, A. Buzo and R. M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Transactions on Communications*, vol. COM-28, pp. 84-95, Jan. 1980.

[17] B. T. Lowerre, *The Harpy Speech Recognition System*, Ph.D. Thesis, Carnegie-Mellon University, 1976.

[18] J. Makhoul, "Linear Prediction: a Tutorial Review," *Proceedings of the IEEE*, vol. 63, no. 4, pp. 561-580, April 1975.

[19] J. D. Markel and A. H. Gray, *Linear Prediction of Speech*, Springer-Verlag, New York, 1976.

[20] C. S. Myers *et al*, "Performance tradeoffs in dynamic time warping algorithms for isolated word recognition," *IEEE Trans. on Acoust., Speech, Signal Processing*, vol. ASSP-28, pp. 622-635, December 1980.

[21] A. M. Noll, "Short-Time Spectrum and Cepstrum Techniques for Vocal-Pitch Detection," *J. Acoust. Soc. Amer.*, 36, 2, pp. 296-302, 1964.

[22] K. C. Pan, F. K. Soong, L. R. Rabiner and A. F. Bergh, "An Efficient Vector-Quantization Preprocessor for Speaker Independent Isolated Word Recognition," *IEEE Int. Conf. Acoust., Speech and Signal Processing*, pp. 23.9.1-23.9.4, April 1985.

[23] T. W. Parsons, *Voice and Speech Processing*, McGraw-Hill, 1986.

[24] D. B. Paul, R. P. Lippmann, Y. Chen, and C. Weinstein, "Robust HMM-Based Techniques for Recognition of Speech Produced under Stress and in Noise," *Speech Tech.*, April 1986.

[25] J. Picone, "Continuous Speech Recognition Using Hidden Markov Models," *IEEE Trans. on Acoust., Speech, and Signal Processing*, pp. 26-41, July 1990.

[26] A. B. Poritz, "Linear predictive hidden Markov models and the speech signal," *IEEE Int. Conf. Acoust., Speech and Signal Processing*, pp. 1291-1294, May 1982.

[27] L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*, Prentice-Hall, Englewood Cliffs, New Jersey, 1978.

[28] L. R. Rabiner and S. E. Levinson, "Isolated and Connected Word Recognition Theory and Selected Applications," *IEEE Trans. on Communications*, vol. COM-29, no. 5, May 1981.

[29] L. R. Rabiner, S. E. Levinson and M. M. Sondhi, "On the Application of Vector Quantization and Hidden Markov Models to Speaker-Independent, Isolated Word Recognition," *Bell Syst. Tech. J.*, vol. 62, no. 4, pp. 1075-1105, April 1983.

[30] L. R. Rabiner, B. H. Pan and F. K. Soong, "On the Performance of Isolated Word Speech Recognizers Using Vector Quantization and Temporal Energy Contours," *AT&T Technical Journal*, 63(7):1245-1260, Sept. 1984.

[31] L. R. Rabiner, B. H. Juang, S. E. Levinson and M. M. Sondhi, "Recognition of Isolated Digits Using Hidden Markov Models With Continuous Mixture Densities," *AT&T Technical Journal*, 64(6):1211-1233, July-August 1985.

[32] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257-285, February 1989.

[33] A. G. Richter, "Modeling of Continuous Speech Observations," *Advances in Speech Processing Conference, IBM Europe Institute*, July 1986.

[34] H. Sakoe and C. Chiba, "Dynamic Programming Algorithm Optimization for Spoken Word Recognition," *IEEE Trans. on Acousi., Speech, and Signal Processing*, vol. ASSP-26, no. 1, pp. 43-49, February 1978.

[35] R. Schwartz, Y. Chow, O. Kimball, S. Roucos, M. Krasner and J. Makhoul, "Context-Dependent Modeling for Acoustic-Phonetic Recognition of Continuous Speech," *IEEE Int. Conf. Acoust., Speech and Signal Processing*, April 1985.

[36] W. C. Scratchley, *A Word-Based Vector Quantization Speech Recognizer with Segmentation*, B.A.Sc. thesis, Simon Fraser University, 1989.

[37] K. Shikano, *Evaluation of LPC Spectral Matching Measures for Phonetic Unit Recognition*, Technical Report, Computer Science Department, CMU, 1986.

[38] J. E. Shore and D. K. Burton, "Discrete Utterance Speech Recognition Without Time Alignment," *IEEE Trans. on Information Theory*, vol. IT-29, no.4, pp. 473-491, July 1983.

[39] A. Tassy and L. Miclet, "Speaker-Independent French Digit Recognition Using Word-Based Vector Quantization and Hidden Markov Models," *in Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, pp. 49.15.1-49.15.4, April 1986.

[40] E. Velez, L. Cossette and V. Cuperman, "Development of a VQ-HMM Continuous Speech Speaker-Independent Recognition System for Small Vocabularies," *in Proc. IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp. 469-472, May 1991.

[41] A. J. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," *IEEE Transactions on Information Theory*, IT-13(2):260-269, April 1967.