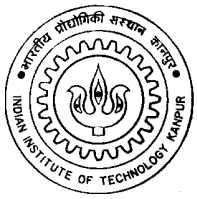
Speaker Independent Isolated Digit Voice Recognition Using Discrete Hidden Markov Model

A Thesis Submitted
in Partial Fulfillment of the Requirements
for the Degree of
Master of Technology

by

K Vasudeva Rao



to the

Department of Computer Science & Engineering
Indian Institute of Technology, Kanpur

July, 2000

Certificate

This is to certify that the work contained in the thesis entitled "*Speaker Independent Isolated Digit Voice Recognition Using Discrete Hidden Markov Model*", by *K Vasudeva Rao*, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

July, 2000

(Dr. Rajat Moona)
Department of Computer Science & Engineering,
Indian Institute of Technology,
Kanpur.

TO

His Divine Grace Srila A. C. Bhakti Vedanta Swami Prabhupada

Who taught me spiritual science by his own practical example and
saved me from the danger of ignorance of purpose of human life

Abstract

Many practical interactive voice response systems require speaker independent speech recognition. Achieving speaker independence is difficult as we do not have direct methods to prepare speaker independent reference patterns of the sub-units of the speech and compare a given sub-unit of speech with them. Hidden Markov Models provide better means than other methods to achieve speaker independence with the help of training speech by a sufficiently large number of speakers. Hidden Markov models have the inherent capability to model the variations in speed of the speech. We developed an interactive voice response system based on discrete Hidden Markov Models. In our system we use a word detector and a linear prediction based signal processing front end which are also developed in this work. We recorded telephone quality speech with the help of modem interface and prepared database of spoken digits of 160 speakers using modem for the training purpose to achieve speaker independence. We also present different fine tuning methods to improve the performance of speech recognition. We also present word rejection criterion to improve confidence of the recognition. We also present an interactive voice response system which is developed using the technology developed in this thesis.

Acknowledgements

I am very much grateful to my guide, Dr. Rajat Moona, whose excellent multi descipline knowledge helped me at every stage of this work. I can not restrain to mention his personal friendly care at times I was dipressed. I am also grateful to Dr. S. Umesh and his Ph.D student Rohit Singh of EE department, Dr. Samudra Vijaya and Dr. P. V. S. Rao of TIFR, Mumbai for their valuable suggestions and comments during thesis discussions. I would also like to thank faculty members of CSE department for gaining better knowledge in different fields of computer science. I would also like to thank Dr. Sumit Ganguly and Dr. T. V. Prabhakar for their interest in my welfare.

This work has been done as a part of industrial interaction program of I. I. T Kanpur with Neomagic Corporation, California, USA. I would like to thank Neomagic Corporation for extending financial support to my thesis and choosing me as "Neomagic Fellow".

I would like to thank whole campus community for their contribution in preparing the spoken digit database. I would also like to thank Oregon Graduate Institute, Oregon for donating their spoken digit database to I. I. T Kanpur to use for academic purposes.

I am greatly indebted to Dr. Manoranjan Sinha, Dr. Kamalesh Kumar Singh, Dr. Sudipto Ghosh and their family members and Madhava Krishna for their hospitality during my stay here. I am also indebted to Dr. Pravat Kumar Giri, Varun, Bhaskar, Vijay, Debashis, Shushovan, Rama Gopal, Prasad, Shekhar, Ajitabh Bharati, Ravi Kumar, Giridhar, Srinivas, Manglik, Subra, Rajan& Co and all other members of Bhaktivedanta club who helped me understanding spiritual science. I am also indebted to my Srikar, Subhash, Gopi, Subhendu, Sridhar, Pradeep, Prasad, Uma, Mandeep, Dwivedi and all other classmates and friends. Finally I would like to thank my parents, brother and sister for their support and encouragement.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

Interactive voice response systems are getting more and more deployed in applications, especially where query traffic is very high and queries are to be attended twenty four hours a day. For these reasons speech recognition has gained lot of interest in the researchers from various fields. Despite this, speech recognition has been one of the most difficult problems to solve. In this work we develop a speaker independent isolated digit recognition system for telephone quality speech. We have used linear prediction, vector quantization and Hidden Markov Model to develop this system. We have collected the required telephone quality speech for training purpose. This speech recognition system uses a modem as input and output device for speech. To interact with the system, the speaker has to dial the telephone number of the modem and interact with it using the telephone.

1.2 Related Work

Research in automatic speech recognition has been done for about five decades. The first speech recognition system was built in the year 1952 at Bell Laboratories. The recognizer was built using acoustic features to recognize the digits spoken by a single speaker. While the research had been carried out with acoustic phonetic approach, in mid 1970s, Itakura showed how linear prediction could be applied to speech recognition [5]. In late 1970s and early 1980s, researchers at AT&T Bell Laboratories

conducted many experiments [10, 11, 16] to incorporate speaker independence in the speech recognition systems. The techniques were refined over a decade. In course of developing an isolated speech speech recognition system, they developed an algorithm for word detection [18]. Although Hidden Markov Model (HMM) was initially introduced in 1960s, researchers at only a few laboratories could apply it to speech recognition after a decade [1, 6]. A decade later it was wide published [15] and became popular. Today almost every speech recognizer uses HMM. Wilpon [17], at AT&T Bell Laboratories studied on ability to automatically recognize the telephone quality speech in real world conditions. He reported a word detection rate of 98% and speech recognition rate of 86% in online digit recognition. He used a total of 11,035 digits of 3100 speakers.

1.3 Goals

- To develop a speaker independent isolated digit voice recognizer for telephone quality speech.
- To built an application for Computer Interactive Voice Response system (CIVRS) that uses the technology developed in this thesis.

1.4 Organization of this work

The rest of the thesis is organized as follows.
In chapter 2, we discuss different approaches to speech recognition and the basic speech recognition system in our implementation. We discuss different parts of speech recognition system namely, signal processing front end, vector quantization and hidden Markov model back-end.
In chapter 3, we present the design and implementation of the speech recognition system. We also present the word-detection algorithm and experimental results that helped us to choose various parameters for the speech recognition system.
In chapter 4, we present various experiments for word rejection criterion and performance fine-tuning.
In chapter 5, we present an interactive voice response system application, developed using the speech recognition technology presented in this thesis. Finally we conclude

this work in chapter 6

Chapter 2

Basic Speech Recognition Techniques

Speech recognition systems accept audio data as input and produce a sequence of symbols corresponding to the sequence of spoken words in the input speech.
Speech signals are slowly varying time signal. When examined over a sufficiently short interval of time (say, 5 to 100 ms), a speech signal is fairly stationary. When examined over a long interval of time, around 200 ms or more, the signal characteristics change to reflect the different sounds spoken. A speech recognition system, therefore, should be able to model the short time characteristics of the signal and their variations over long periods of time.
Even though extensive research has been carried out during the past five decades, we are far from achieving the goal of a robust speech recognizer which can understand spoken words on any subject by all speakers in all environments. Following are some of the reasons for the difficulty.

- Lack of a sophisticated and yet tractable model of speech.
- Differences in the vocal tract sizes among individual speakers contribute to the variability of speech and most of the parametric representations of speech are not completely speaker independent.
- Inherent mismatch between training and test environments.
- Lack of consistent units of speech that are trainable and relatively insensitive to context.

- Inadequate use of human knowledge of acoustics and phonetics.

Several speaker dependent recognition systems are available with acceptable performance. Achieving speaker independence has been the most difficult task in realizing the speech recognition systems. This is due to the speaker dependent nature of parametric representations of speech, and a set of reference patterns suitable for one speaker may perform poorly for another speaker.

There are three approaches to achieve speaker independence. The first approach is to find the perceptually motivated speech parameters that are relatively invariant among speakers. The second approach is to use multiple representations for each speech unit to capture the between-speaker variations. In this approach, for each speech unit we have a very large database. Using this database, a model for each speech unit is generated. During recognition speech-unit models for various speech units are used for comparison. In the third approach, the recognizer knows various characteristics of the speaker after a few sentences and uses this knowledge to adapt the system to the new speaker. Adaptation starts with an initial set of parameters. The new speaker is asked to speak known sentences and the response is used to to adjust the set of parameters.

In this work, the first two approaches are incorporated up to some extent. The speech processing front-end generates differential cepstral coefficients. These coefficients incorporate the formant slope information which is relatively invariant among speakers. The back-end of the speech recognition uses Hidden Markov Model (HMM), which incorporates several reference reference patterns for a speech unit.

In this chapter different parts of the speech recognition system are discussed that are implemented in this work. Among the different parts are Linear Predictive Coding (LPC) model, Vector Quantization and Hidden Markov Model.

2.1 Approaches to Speech recognition

Speech recognition approaches can be broadly classified into three categories [14].

1. Acoustic phonetic approach
2. Pattern recognition approach
3. Artificial intelligence approach

2.1.1 Acoustic Phonetic Approach

The Acoustic phonetic approach is based on the theory of acoustic phonetics with the assumption that there exist finite, distinctive phonetic units in the spoken language and these units can be broadly characterized by a set of properties. The acoustic properties of the phonetic units are highly variable, both with speakers and with other phonetic units. It is assumed that the rules governing these variations are straight forward and can be learned and applied in practical situations.

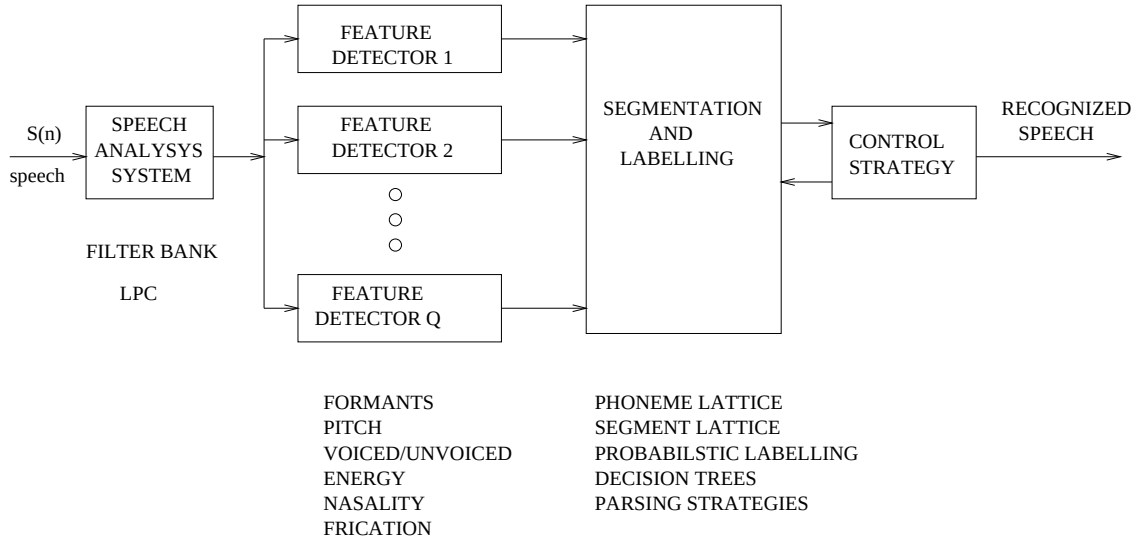


Figure 2.1: An Acoustic Phonetic Speech Recognition System

The first task in this method is to segment the input speech into discrete regions. Each of these regions corresponds to one or more phonemes. For this we analyze the speech in short intervals of time to study the spectral properties and then characterize these spectral properties as acoustic events, such as formants, pitch, nasality, frication, etc. (Figure 2.1). Using this acoustic event information, we label the discrete regions as one or more possible phonemes. The exact sequence of phonemes in the speech is however not known at this stage. This is because some regions might have been labeled as more than one possible phonemes. The exact combination of phonemes in the speech is determined by the dictionary of words with their phoneme sequence. The grammar of the language also plays an important role in this process.

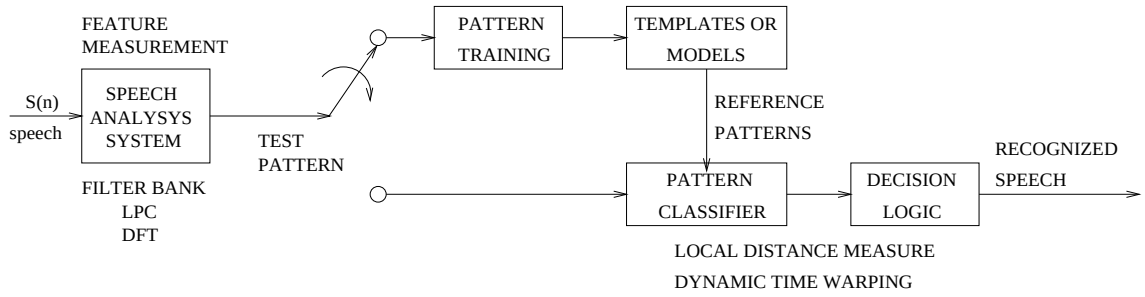


Figure 2.2: Pattern Recognition Speech Recognition System

2.1.2 Pattern Recognition Approach

In the pattern recognition approach, speech patterns are used directly without explicit acoustic characterization of the speech spectral analysis information. This approach generally has two stages, namely speech pattern training and speech pattern recognition. The concept is that if enough versions of a pattern to be recognized are included in a training set, then the system should be able to adequately characterize the acoustic properties of the pattern. During pattern training, the system learns acoustic properties of the speech class, reliable and repeatable for the given training set. In pattern recognition stage, the system compares an unknown speech with all trained patterns, and classifies the unknown speech according to the goodness of the match.

2.1.3 Artificial Intelligence Approach

The basic idea of artificial intelligence approach is to compile and incorporate knowledge from variety of sources to realize the different stages of speech recognition system. This approach is a hybrid of the acoustic-phonetic approach and the pattern recognition approach. It exploits the ideas and concepts of both methods and attempts to mechanize the recognition procedure according to the way a person applies his intelligence. The following are some of the knowledge sources and their brief description.

- Acoustic knowledge: Evidence of which phonetic units are spoken on the basis of spectral measurements and presence or absence of features.
- Lexical knowledge: The combination of acoustic evidences so as to postulate word as specified by a lexicon that maps sounds into words.

- Syntactic knowledge: The combination of words to form the grammatically correct strings.
- Semantic knowledge: Understanding of the task domain so as to be able to validate sentences and phrases that are consistent with the task being performed, and the previously decoded sentences.
- Pragmatic knowledge: Inference ability necessary in resolving ambiguity of meaning based on ways in which words are generally used.

In this thesis we have implemented the second approach, the pattern recognition approach.

2.2 Liner Predictve Coefficients (LPC) Model for Speech Recognition

The first task of the Pattern recognition approach is to parametrically represent the speech signal. Among the different possibilities to represent the speech parametrically, spectral envelop for short duration of the speech is probably the best. The function of feature measurement block in figure 2.2 is to represent the speech signal in terms of compact, efficient set of speech parameters.

2.2.1 Introduction

The theory of linear prediction [13] as applied to speech, has been well understood for many years. Following are some of the reasons underlying the widespread usage of LPC.

1. LPC provides a good model of speech signal and provides a good approximation to the vocal tract spectral envelop. During the unvoiced and the transient regions of speech, the LPC model is less effective than for the voice speech, but it still provides an acceptably useful model for speech-recognition purposes.
2. The way in which LPC is applied to the analysis of speech signals leads to a reasonable source-vocal tract separation. This is important for a speaker independent voice recognition system.

3. LPC is an analytically tractable model. The method of LPC is mathematically precise and is simple and straightforward to implement.

4. The LPC model has been shown to work well in voice recognition applications [4, 14].

2.2.2 The Model

In linear prediction, a sample in the signal is predicted as a linear combination of its past values. Let the predicted time series of the signal be $\hat{s}_1$, $\hat{s}_2, \cdots, \hat{s}_n$ and the real signal be s_1, $s_2, \cdots, s_n$, then the predicted value $\hat{s}_i$ of the i^{th} sample is a linear combination of (s_{i-1}, s_{i-2}, $\cdots, s_{i-p}$), where p is called the order of the LPC model. In LPC model, with an order p, only p coefficients are needed. These coefficients can be computed by solving a set of equations which are well documented in the standard texts [13, 19]. This LPC coefficient vector is spectral approximation of the speech signal.

2.3 Vector Quantization

2.3.1 Introduction

The LPC coefficient vectors are generated for the waveform on a short time basis. These vectors are however very large. A technique of vector quantization helps in reducing this space. In vector quantization, a table called code book of finite size is maintained. Each entry of the code book is a vector. The spectral LPC vectors are then mapped on to one of these vectors and only the index is used to represent the waveform instead of the entire LPC vector.
For vector quantization, it is necessary to have a measurement of dissimilarity between the two vectors. We expect such dissimilarity measure to confirm to the known linguistic characteristics.

2.3.2 Distance Measure Criterion

Let x, y be two feature vectors defined on a vector space χ. We define a metric or distance function d on the vector space χ as a real-valued function with the following

properties.

1. *Positive definiteness:* $0 \leq d(x, y) < \infty$ for $x, y \in \chi$ and $d(x, y) = 0$ if and only if $x = y$

2. *Symmetry:* $d(x, y) = d(y, x)$ for $x, y \in \chi$

3. *Triangle inequality:* $d(x, y) \leq d(x, z) + d(y, z)$ for $x, y, z \in \chi$

4. *Invariance:* $d(x + z, y + z) = d(x, y)$

2.3.3 VQ training and classification structure

To build a VQ codebook and implement a VQ analysis procedure, we need the following:

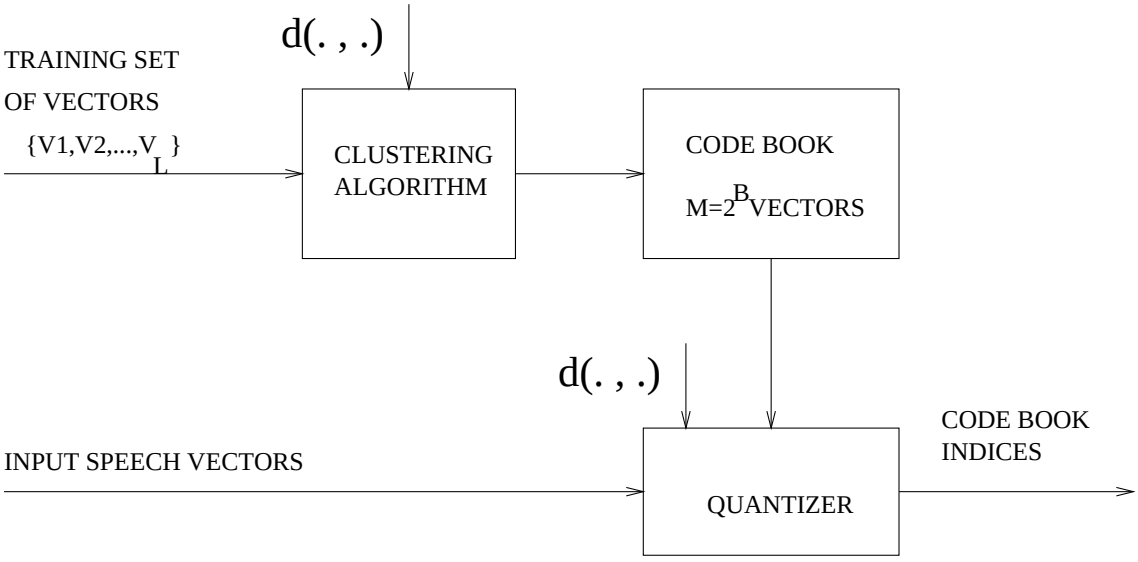


Figure 2.3: Vector quantization training and classification structure

1. A large *set of spectral vectors*, $v_1, v_2, \cdots, v_L$, which form a training set. The training set is used to create the optimal set of code book vectors for representing the spectral variability observed in the training set.

2. A *distance measure* between a pair of spectral analysis vectors so as to be able to cluster the training set vectors as well as to classify arbitrary spectral vectors into unique code book entries.

3. A *centroid computation* procedure. On the basis of the partitioning that classifies the training vectors into the M clusters, we choose the M codebook vectors as the centroid of each of the M clusters.

4. A *classification procedure* for arbitrary speech spectral analysis vectors that chooses codebook vector closest to the input vector and uses the codebook index as the resulting spectral representation.

2.3.4 Clustering the Training Vectors

The way in which set of L training vectors can be clustered into set of M codebook vectors is as follows.

- *K-means Clustering Algorithm* [14]

    1. Initialization: Arbitrarily chose M training vectors as the initial set of code words in the codebook.
    2. Nearest-Neighbor Search: For each training vector, find the codeword in the current codebook that is closest as measured by the distance measure, and assign that vector to the corresponding cluster, associated with closest code word.
    3. Centroid Update: Update the code word in each cluster using the centroid of the training vectors assigned to that cluster.
    4. Iteration: Repeat the steps 2 and 3 until the relative of average distance of all training vectors to their corresponding code word falls bellow some threshold.

- *Binary Split Algorithm* [14]
  Instead of starting directly with the M initial arbitrary training vectors, binary split algorithm starts with one initial vector and achieves codebook of size M after $log_2 M$ steps. At each step it applies *K-means clustering algorithm* to achieve the optimum codebook entries. Then it splits each codebook vector into two, giving double the size of present codebook entries. These newly split entries are used as initial vectors for the next step.

2.4 The Hidden Markov Model

Hidden Markov Model (HMM) [15] approach is a widely used statistical method for characterizing the spectral properties of the frames of a pattern. The underlying assumption of HMM is that the speech signal can be well characterized as a parametric random process, and that the parameters of the stochastic process can be determined (estimated) in a precise and well defined manner.

2.4.1 Definition

Hidden Markov Model is a doubly embedded stochastic process with an underlying stochastic process that is not directly observable (it is hidden) but can be observed only through another set of stochastic processes that produce the sequence of observations.
A hidden Markov model is collection of states connected by transitions. Each state carries two sets of probabilities: a set of transition probabilities, which provides the probabilities of transitions from this state to all the states; and output probabilies which define the conditional probability of emitting each output symbol if the system is in that state. Figure 2.4 shows an example of a HMM with two output symbols, A and B.

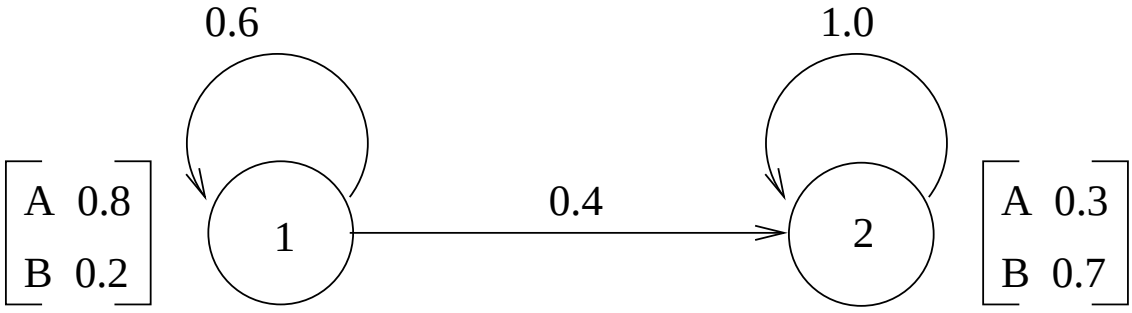


Figure 2.4: A simple HMM with two states and and two output symbols, A and B

Given an observation sequence (a string of A's and B's for the example in figure 2.4) generated by a HMM, we however cannot determine the exact state transition sequence. This is because in each state the output symbol is not unique (in contrast with the Markov chain). The output symbol is again a random variable. So, for a given observation sequence, we cannot see the underlying process. Hence this model

has the Hidden Markov Model.
A HMM is characterized by the following:

1. N, the number of states in the model. We denote the individual states as $S = \{S_1, S_2, \cdots, S_n\}$, and the system state at time t as q_t

2. M, the number of distinct observation symbols per state, i.e., the discrete alphabet size. We denote the individual symbols as $V = \{v_1, v_2, \cdots, v_m\}$.

3. The transition probability distribution $A = \{a_{ij}\}$, where each a_{ij} is the transition probability from state S_i to state S_j. Clearly, $a_{ij} \geq 0, \quad \forall i, j$ and $\sum_j a_{ij} = 1, \quad \forall i$

4. The observation symbol probability distribution $B = b_{jk}$, where each b_{jk} is the observation symbol probability for symbol v_k, when the system is in the state S_j. Clearly, $b_{jk} \geq 0, \quad \forall j, k$ and $\sum_k b_{jk} = 1, \quad \forall j$

5. The initial state distribution $\pi = \{\pi_i\}$, where $\pi_i = P[q_1 = S_i], \quad 1 \leq j \leq N$

We can specify an HMM model as $\lambda = (A, B, \pi, M, N, V)$. In this thesis we however represent $\lambda = (A, B, \pi)$ and assume M, N and V to be implicit.

2.4.2 Use of HMM in Speech Recognition

HMM can be used to model a unit of speech, whether it is a phoneme, or a word, or a sentence. LPC analysis followed by the vector quantization of the unit of speech, gives a sequence of symbols (VQ indices). HMM is one of the ways to capture the structure in this sequence of symbols. In order to use HMMs in speech recognition, we should have some means to achieve the following.

- *Evaluation:* Given the observation sequence $O = o_1, o_2, \cdots, o_T$, and a HMM $\lambda = (A, B, \pi)$, to efficiently compute $P(O|\lambda)$, the probability of the observation given the HMM.

- *Decoding:* Given the observation sequence $O = o_1, o_2, \cdots, o_T$, and a HMM $\lambda = (A, B, \pi)$, to choose a corresponding state sequence $Q = q_1, q_2, \cdots, q_T$ which is optimal in some meaningful sense, given the HMM.

- *Training:* To adjust the HMM parameters $\lambda = (A, B, \pi)$ to maximize $P(O|\lambda)$.

The following are some of the assumptions in the hidden Markov modeling for speech.

- Successive observations (frames of speech) are independent, and therefore the probability of sequence of observations $P(o_1, o_2, \cdots, o_T)$ can be written as a product of probabilities of individual observations, i.e.,

$$P(o_1, o_2, \cdots, o_T) = \Pi_{i=1}^{T} P(o_i)$$

- *Markov assumption:*The probability of being in a state at time t, depends only on the state at time t-1

Evaluation

Evaluation is to find probability of generation of a given observation sequence by a given model. The recognition result will be the speech unit corresponding to the model that best matches among the different competing models. Now we have to find $P(O|\lambda)$, the probability of observation sequence $O = (o_1, o_2, \cdots, o_T)$ given the model λ i.e., $P(O|\lambda)$.
One could, in principle compute $P(O|\lambda)$ by computing the joint probability, $P(O, q|M)$ for each possible state sequence, q, of length T and then summing over all state sequences. Computationally this method is very costly. However, there is an efficient way of computing this probability using *forward* and *backward* probabilities.

Forward-Backward Algorithm [15]

- *The Forward Probabilities:* consider the forward variable $\alpha_t(i)$ defined as

$$\alpha_t(i) = P(o_1 o_2 \cdots o_t, q_t = i|\lambda) \tag{1}$$

that is, the probability of the partial observation sequence,$o_1\ o_2\ \cdots o_t$ (until time t) and state i at time t, given the model λ. We can solve for α_t inductively, as follows:

1. *Initialization:*

$$\alpha_1(i) = \pi_i b_i(o_1) \quad 1 \leq i \leq N \tag{2}$$

2. *Induction :*

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(o_{t+1}) \quad 1 \leq t \leq T-1,\ 1 \leq j \leq N \tag{3}$$

where N is the number of states in the model.

- *The Backward Probabilities:* Consider the backward variable $\beta_t(i)$ defined as

$$\beta_t(i) = P(o_{t+1}o_{t+2}\cdots o_T|q_t = i, \lambda) \tag{4}$$

that is, the probability of the partial observation sequence from $t+1$ to the end, given state i at time t and the model λ. Again, we can solve for $\beta_t(i)$ inductively as follows

1. *Initialization:*

$$\beta_T(i) = 1, \quad 1 \leq i \leq N \tag{5}$$

2. *Induction:*

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j), \tag{6}$$
$$t = T-1, T-2, \cdots, 1, \quad 1 \leq i \leq N$$

The two *forward* and *backward* probabilities can be used to compute $P(O|\lambda)$ according to

$$P(O|\lambda) = \sum_{i=1}^{n} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \tag{7}$$

for any t such that $1 \leq t \leq T-1$. Equations (3) to (7) are referred to as *forward-backward algorithm.*
Setting $t = T-1$ in (7) gives

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{8}$$

so that the probability, $P(O|\lambda)$ can be calculated form *forward probabilities* alone. Similarly $P(O|\lambda)$ can be calculated from *backward probabilities* alone, by setting $t = 1$.

$$P(O|\lambda) = \sum_{i=1}^{N} \pi_i b_i(o_1) \beta_1(i) \tag{9}$$

Decoding

Decoding is to find the single best state sequence, $Q = (q_1, q_2 \cdots q_T)$, for the given observation sequence $O = (o_1 o_2 \cdots o_T)$. Consider $\delta_t(i)$, defined as

$$\delta_t(i) = \max_{q_1, q_2, \cdots q_{t-1}} P[q_1 q_2 \cdots q_t = i, o_1 o_2 \cdots o_t|\lambda] \tag{10}$$

that, is $\delta_t(i)$ is the best score along single path, at time t, which accounts for the t observations and ends in state i. By induction, we have

$$\delta_{t+1}(j) = \left[\max_i \delta_t(i) a_{ij}\right] b_j(o_{t+1}) \tag{11}$$

The complete procedure is as follows

The Viterbi Algorithm [15]

1. *Preprocessing:*

$$\begin{aligned} \tilde{\pi}_i &= \log(\pi_i), && 1 \le i \le N \\ \tilde{b}_i(o_t) &= \log\left[b_i(o_t)\right], && 1 \le i \le N,\ 1 \le t \le T \\ \tilde{a}_{ij} &= \log(a_{ij}), && 1 \le i,\ j \le N \end{aligned}$$

2. *Initialization:*

$$\begin{aligned} \tilde{\delta}_1(i) = \log(\delta_1(i)) = \tilde{\pi}_i + \tilde{b}_i(o_1), && 1 \le i \le N \\ \psi_1(i) = 0, && 1 \le i \le N \end{aligned}$$

3. *Recursion*

$$\begin{aligned} \tilde{\delta}_t(j) = \log(\delta_t(j)) = \max_{1\le i\le N}\left[\tilde{\delta}_{t-1}(i) + \tilde{a}_{ij}\right] + \tilde{b}_i(o_1) \\ \psi_t(j) = \arg\max_{1\le i\le N}\left[\tilde{\delta}_{t-1}(i) + \tilde{a}_{ij}\right] \qquad 2 \le t \le T, 1 \le j \le N \end{aligned}$$

4. *Termination*

$$\begin{aligned} \tilde{P}^* &= \max_{1\le i\le N}\left[\tilde{\delta}_T(i)\right] \\ q_T^* &= \arg\max_{1\le i\le N}\left[\tilde{\delta}_T(i)\right] \end{aligned}$$

5. *Backtracking*

$$q_t^* = \psi_{t+1}(q_{t+1}^*), t = T-1, T-2, \cdots, 1$$

The array q^* contains the required best state sequence.

Learning

Learning is to adjust the model parameters (A, B, π) to maximize the probability of the observation sequence given the model. It is the most difficult task of the Hidden Markov Modeling, as there is no known analytical method to solve for the parameters in a maximum likelihood model. Instead, an iterative procedure should be used. *Baum-Welch algorithm* [15] is the extensively used iterative procedure for choosing the model parameters. In this method we start with some initial estimates of the model parameters and modify the model parameters to maximize the training observation sequence in an iterative manner till the model parameters reach a critical value. We define the variables $\xi_t(i,\, j)$ and $\gamma_t(i)$ as,
$\xi_t(i,\, j)$ is the probability of being in state i at time t, and state j at time $t+1$, given the model and observation sequence.

$$\xi_t(i,\, j) = P(q_t = i, q_{t+1} = j | O, \lambda)$$

$$\xi_t(i,\, j) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)} \tag{12}$$

where $\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)$ is equal to $P(q_t = i, q_{t+1} = j, O|\lambda)$.
$\gamma_t(i)$ is the probability of being in the state i at time t, given the observation sequence O, and the model λ

$$\gamma_t(i) = P(q_t = i | O, \lambda)$$

$$\gamma_t(i) = \frac{\alpha_t(i) \beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i) \beta_t(i)} \tag{13}$$

where $\alpha_t(i)\beta_t(i)$ is equal to $P(O,\, q_t = i|\lambda)$
By the definition of the variables $\xi_t(i,\, j)$ and $\gamma_t(i)$, the following relations are true,

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,\, j)$$

$$\sum_{t=1}^{T-1} \gamma_t(i) = \textit{ expected number of transitions from state } i \textit{ in } O$$

$$\sum_{t=1}^{T-1} \xi_t(i,\, j) = \textit{ expected number of transitions from state } i \textit{ to state } j \textit{ in } O$$

Using the above formulas and the concept of counting the event occurrence, the parameters of the model $\lambda = (A, B, \pi)$ can be re-estimated as $\tilde{\lambda} = (\tilde{A}, \tilde{B}, \tilde{\pi})$, as

follows.

$$\tilde{\pi}_i = \gamma_1(i) \tag{14}$$

$$\tilde{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,\, j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{15}$$

$$\tilde{b}_j(k) = \frac{\sum_{t=1\ s.t\ o_t=v_k}^{T-1} \gamma_t(i)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{16}$$

It has been proved that one of the following two statements is true for λ and $\tilde{\lambda}$

1. The initial model λ defines a critical point of the likelihood function in which case λ=$\tilde{\lambda}$.

2. model $\tilde{\lambda}$ is more likely than the model λ, in the sense that $P(O|\tilde{\lambda}) > P(O|\lambda)$.

In case (1), we stop the iterative procedure declaring λ as the final trained model for the observation sequence O. In case (2), we replace the the model λ by $\tilde{\lambda}$ as the initial model for the next iteration. The iteration is stopped when $\frac{P(O|\tilde{\lambda})-P(O|\lambda)}{P(O|\lambda)}$ reaches some minimum value and then the model, $\tilde{\lambda}$ is declared as the final trained model for the observation sequence O.

Chapter 3

Design and Implementation

In this chapter we discuss the design and implementation issues of realizing each stage of our basic recognizer. We also present relevant experimental results where ever they are required for justification and understanding.

3.1 Word End Points Detection

For isolated word recognition, it is assumed that the the words are spoken with a sufficient pause so that no two successive words overlap with each another. Given an input speech, the problem of finding the word boundaries in the time domain is word-detection problem. Word boundaries are detected by characterization of the energy changes over time. In our work, the end-point detection algorithm is similar to the one given by Wilpon etc. [18]. However we have evolved the algorithm by the experience of manual editing of the end points and by different trial and error experiments with the speech of 100 speakers. Figure 3.1 is the speech signal of a typical speaker sampled at the rate of $7.2\,KHz$. The next figure 3.2 is the plot of the energy of the sliding frame of the corresponding signal. The size of the frame is 300 samples and the slide is 100 samples. Starting with the initial 300 samples as the first frame, at every step energy of the frame is calculated and the frame is moved 100 samples forward. This way the energy is plotted for the speech signal in figure 3.2 and others.

The speech signal in figure 3.1 contains five isolated words, which are the spoken

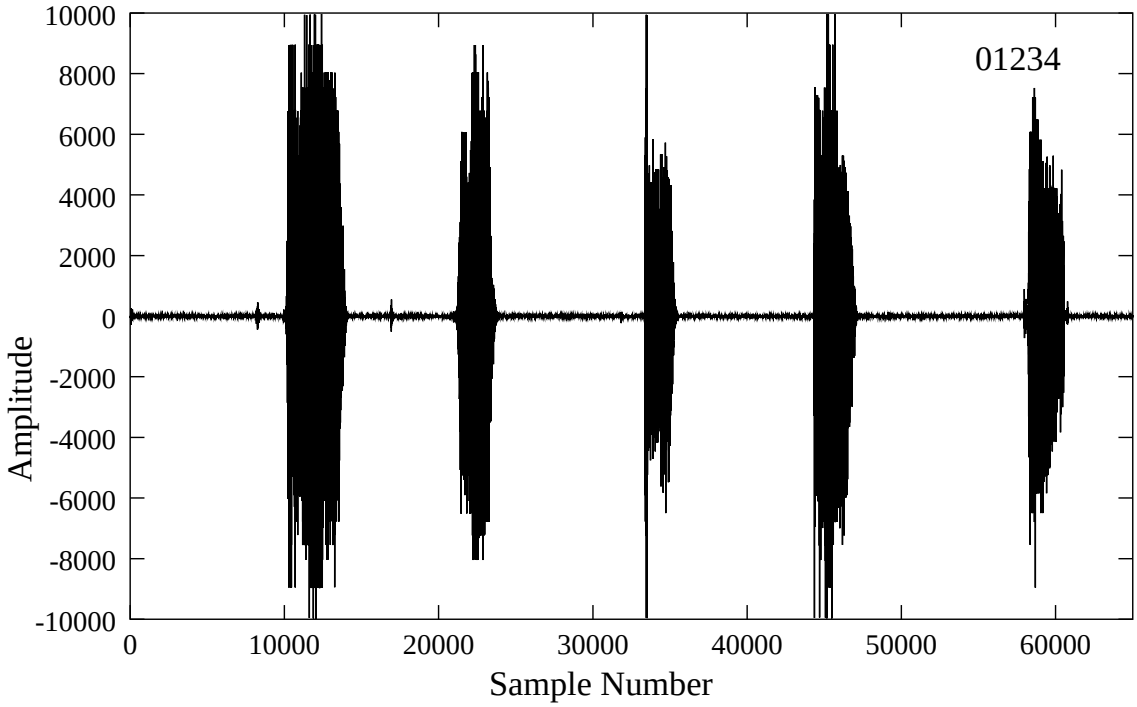


Figure 3.1: Telephone quality Speech signal of a speaker sampled at 7.2 KHz for digits 0, 1, 2, 3, 4

digits 0, 1, 2, 3 and 4. The small spikes are due to the presence of noise. By correlating the speech signal and its corresponding energy plot we can easily guess that the energy can be used as a feature to determine the word boundaries. To start with, we can formulate the algorithm to detect a word as a group of consecutive frames where the log of energy of each frame is above a threshold. While a high threshold can be used to avoid noise, some end points will not be recognized. Similarly a low threshold may result in recognizing noise as a word. To resolve this, a minimum number of frames criteria is introduced in-order to qualify a group of consecutive frames as a word. Though the duration of noise is small in the case of noise like mouth click and switching, it is likely to be long enough to qualify as a word. An example is long heavy breathing noise. To deal with such type of cases, a second higher level of threshold is introduced. The observation is that in most of the cases the maximum energy of a frame in the noise is much less than the maximum energy of a frame of any spoken words unless the background noise is as strong as the speech itself. With this new additional criteria, it is possible to detect the word close to the real end points unless the noise itself is as strong as the speech.

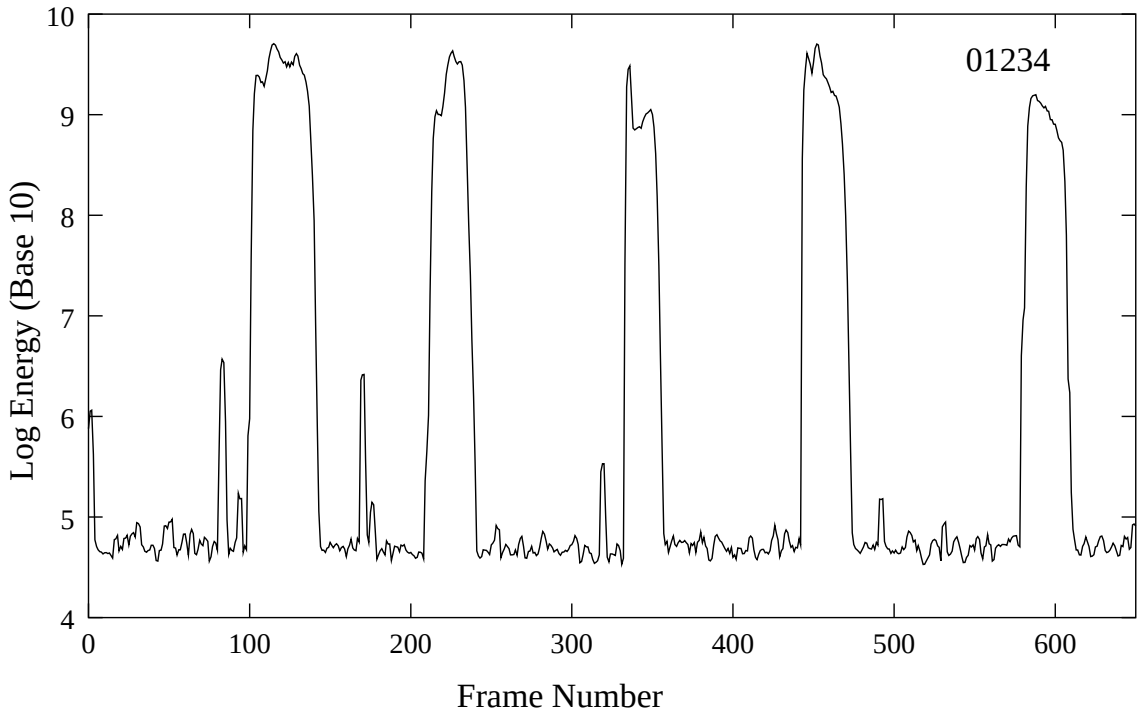


Figure 3.2: Energy of the sliding frame of size=300 samples, sliding size=100 samples over the speech signal in figure 3.1

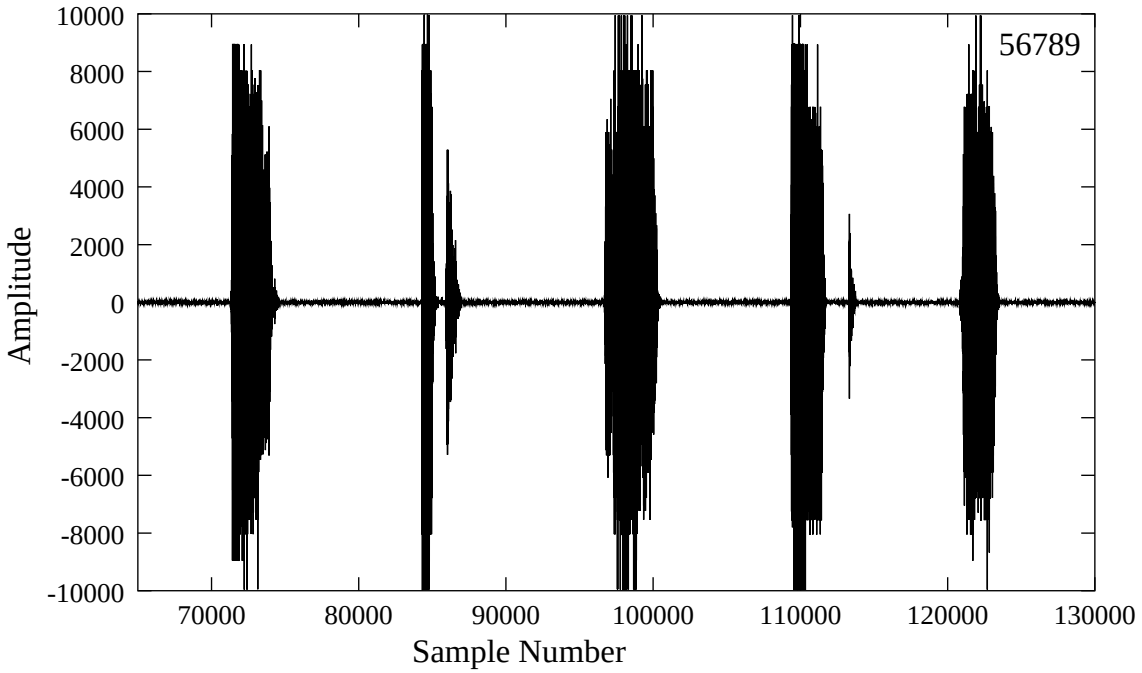


Figure 3.3: Telephone quality Speech signal of the same speaker sampled at 7.2 KHz for digits 5, 6, 7, 8, 9

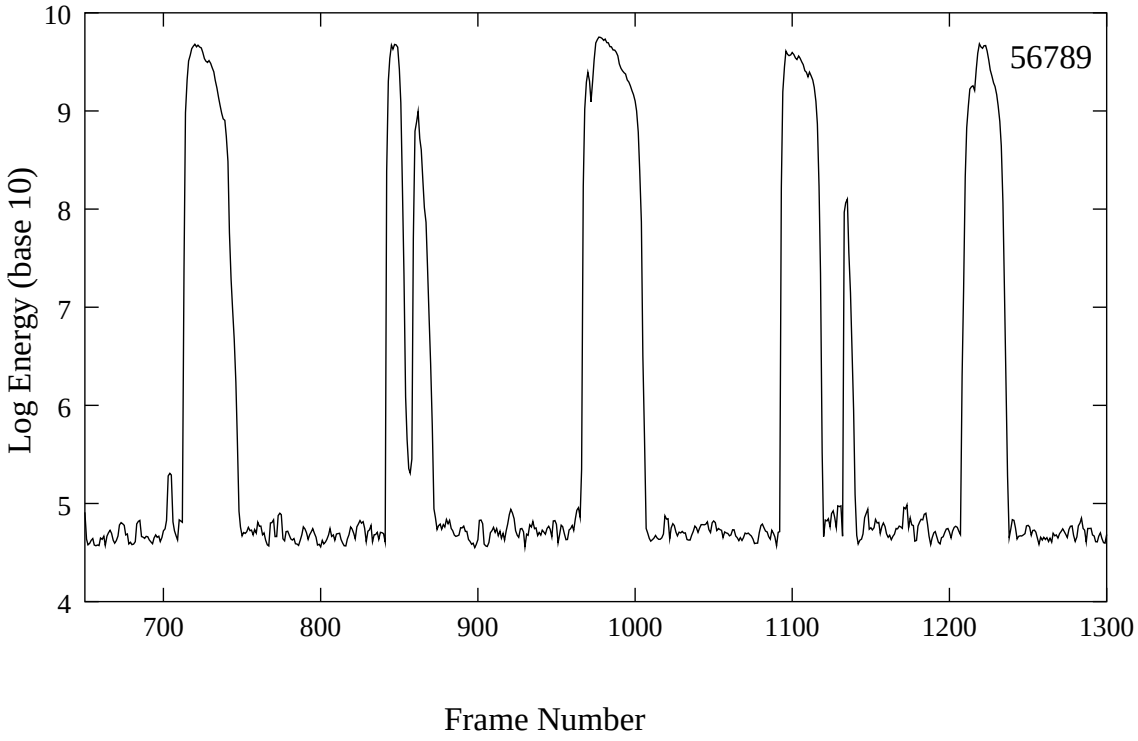


Figure 3.4: Energy of the sliding frame of size=300 samples, sliding size=100 samples over the speech in figure 3.3

However this criterion fails in some cases, for example, consider the speech signal and corresponding frame energy plot in the figures 3.3 and 3.4. These figures correspond to the digits 5, 6, 7, 8 and 9 respectively. In the figure 3.4, two energy bands that appear between the frame numbers 800 and 900 correspond to the digit *six* and the other two energy bands that appear between the frame numbers 1050 and 1150 correspond to the spoken digit *eight*. The second thinner band in these two pairs correspond to 'IX' of the spoken digit *six* and 'T' of the spoken digit *eight* respectively. Under our criterion, these small bands will be dropped or at best recognized as a separate word. However, these portions in the digits *six* and *eight* are critical for good recognition rate. In our experiments in about 50% of the cases these two digits appear as two different bands in the energy plot. The number of frames between the two successive bands of the same word is considerably less than that between two consecutive words. Addition of this heuristic in our criterion works very well. In our experiment, with this new heuristic, 'IX' portion of the spoken digit *six* is dropped in one amongst 90 speakers and 'T' portion of the spoken digit *eight* is dropped for 4 amongst 90 speakers. In our algorithm, we restricted the number of energy pulses

within a single utterance to a maximum of two, which is good enough in case of digit recognition. Out of 873 digits ospoken by 90 speakers, 17 digits were spoken in connected fashion, and were not isolated in the speech. Out of 856 isolated digits, 844 digits were detected without any type of error, 6 digits were detected but the end points were not placed exactly and 6 digits were not detected. The total number of digits that are detected are 850 out of 856, which is 99.3% . These undetected words are mainly due to very low volume at which they were spoken.
Figures 3.5, 3.6, and 3.7 depict the finite state machine which implements the word detection algorithm. The algorithm is designed in such a way that both the word-detection and recognition can go simultaneously running in two different processes. As soon as the word-detection process enters the state 4, it can start sending the frames to the Recognition process for further processing anticipating it to be a word. Later when the word-detection process finds a wrong detection of the word, it can then notify the recognition process to either retain it or discard it. This design is due to our initial mind-set to let word-detection and rest of the recognition to go on parallel. Authors J. G. Wilpon, L. R. Rabiner in their article [18], propose to examine the whole speech of that particular recording to determine minimum energy level frame as the mean noise level and then depending on that set the first and second level thresholds of energy. Since we have decided that our algorithm should detect the words online, the first level threshold is determined prior to the start of the recording. For this we examined the speech containing around 2300 spoken digits, and then fixed the first and the second levels of thresholds. In order to qualify a group of consecutive frames, in our algorithm, at least four consecutive frames should cross the second level threshold. J.G.Wilpon etc.[17] reported word-detection rate of 98%, as opposed to ours (99.3%).

State Diagram for the Word-detection Algorithm

Figures 3.5 to 3.6 depict the finite state diagram for the word-detection algorithm described above. Here we explain the terms used in the finite state diagram and a brief explanation of it.

- FC : Global frame count
- E : Energy of the current frame
- PE : Energy of the previous frame

- *bar* : It is used alternatively for the term pulse
- *LthCFC* : Low threshold consecutive frame count
- *HthCFC* : High threshold consecutive frame count
- *MaxHthCFC* : Maximum High threshold consecutive frame count within a word
- *LThE* : Low threshold energy
- *HThE* : High threshold energy
- *ThickBarLThE* : LThE for Thick energy pulse
- *ThinBarLThE* : LThE for Thin energy pulse
- *WordPending* : if TRUE, this flag indicates that a thick pulse has been recognized as word and waiting to recognize the end point. This indicates that the algorithm is looking for thin pulse, within the allowed scope, to determine the end point

The following is the brief explanation of the state diagram.

- The statements under each state are executed by the system in that state
- The loop covering the states 4 and 5 counts the number of low threshold consecutive frame count
- The loop covering the states 7, 8 and 9 counts the number of high threshold consecutive frame count
- states $2a$ and $2b$ controls which thresholds to be used by the counting loops. This is because we are using different low and high thresholds for thick and thin energy bands.
- The system will be in states 1 and $2*$ when the frame energies are below both the thresholds.
- State $2b$ collect the word boundaries if the thin energy band is not found within the specified scope.
- State $20a$ collects the word boundaries when a thin band also qualifies its threshold energy frame count criterion. This state restricts number of thin bands to a maximum of one.

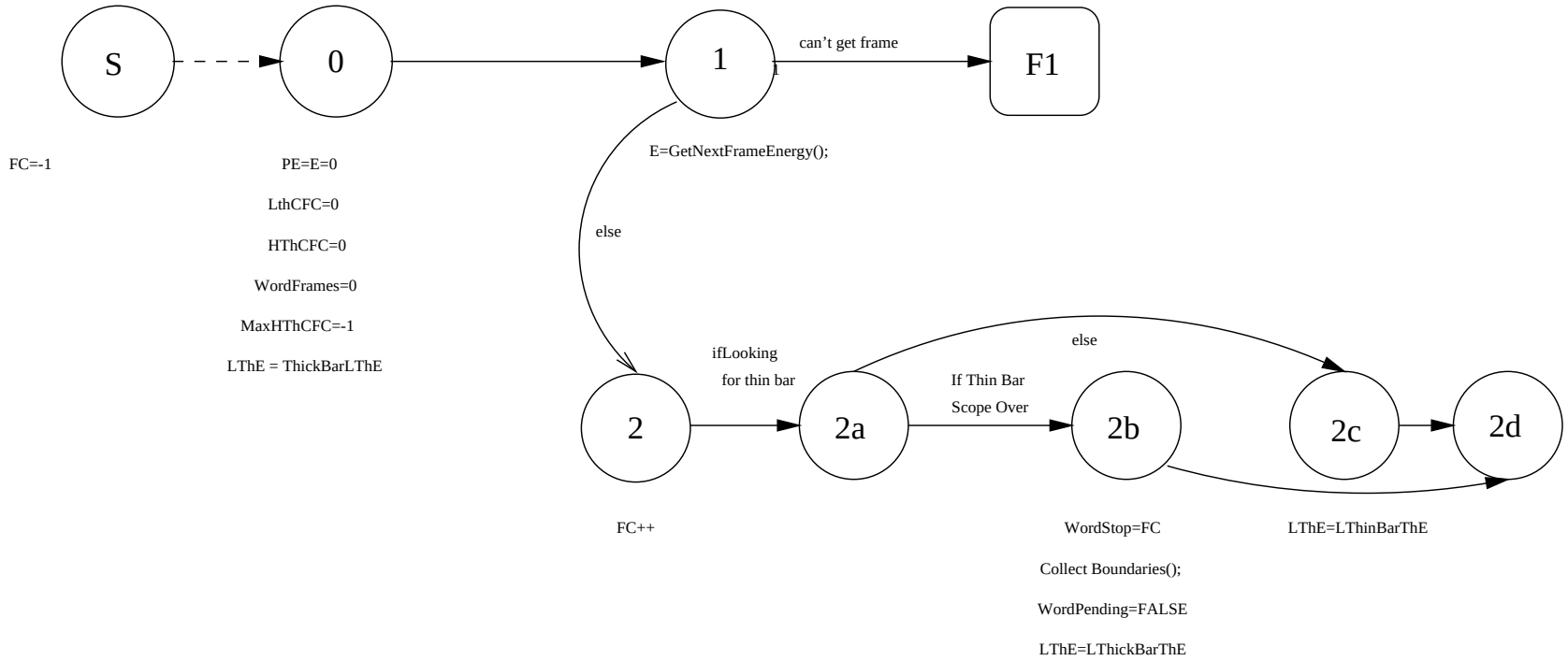


Figure 3.5: Finite State Diagram of Word detection algorithm

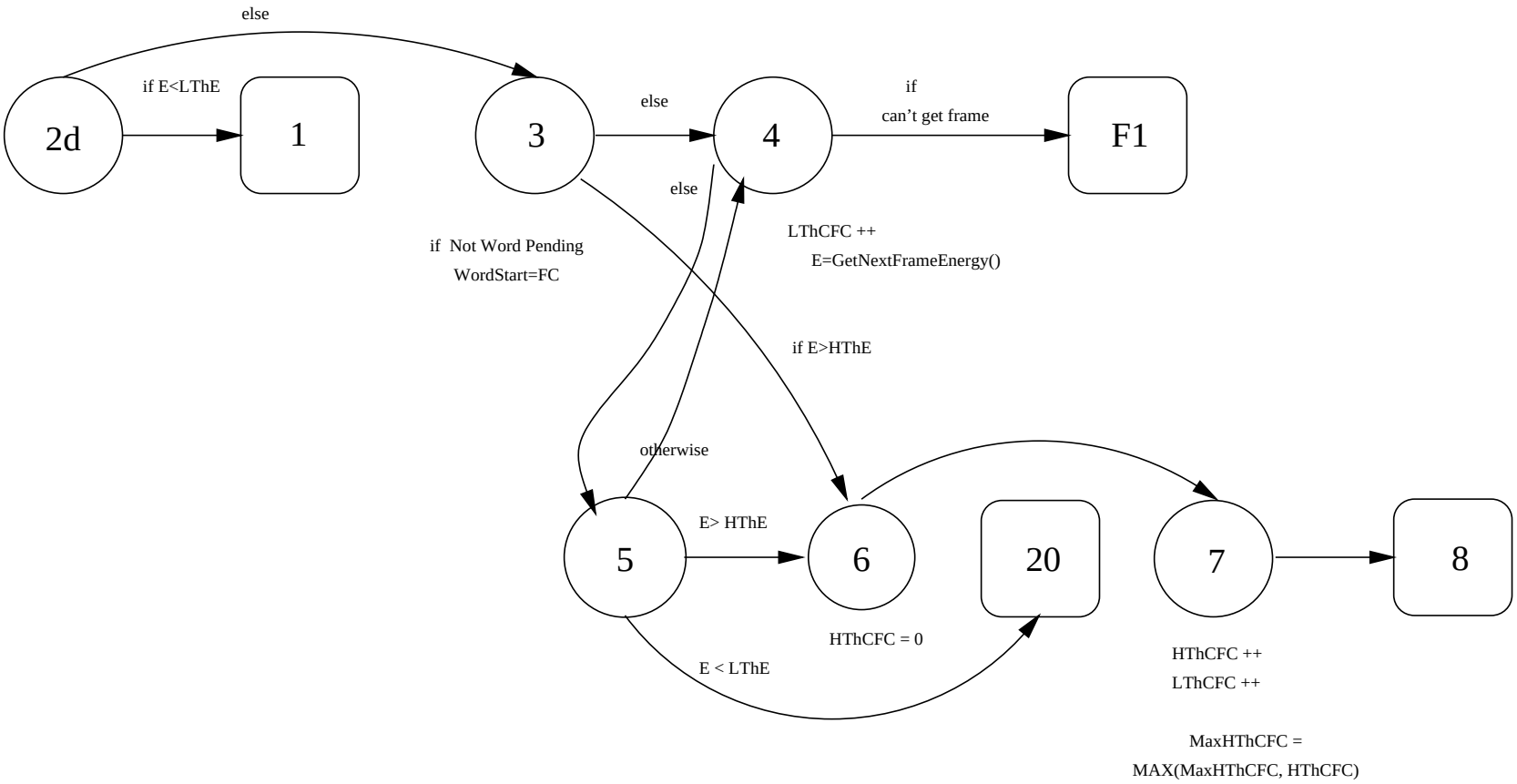


Figure 3.6: Finite State Diagram of Word detection algorithm (cont’d. from 3.5)

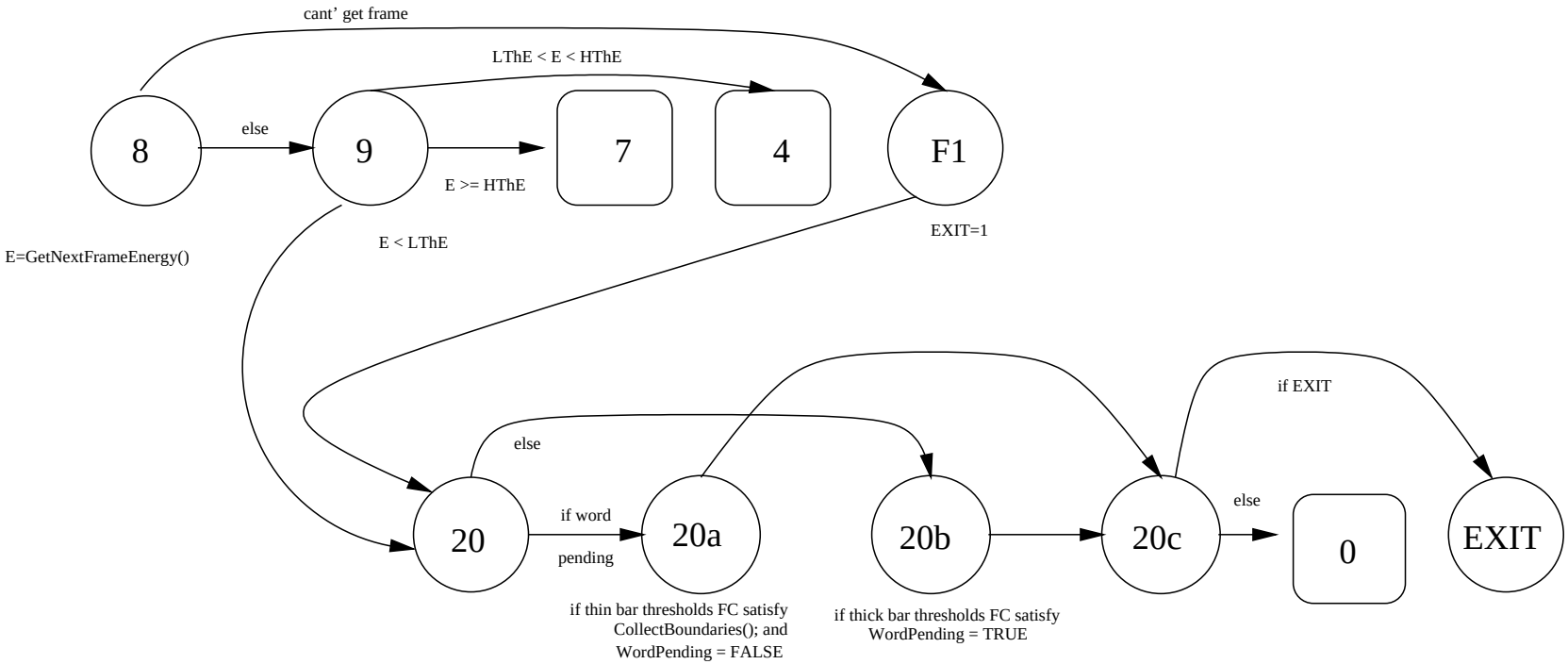


Figure 3.7: Finite State Diagram of Word detection algorithm (cont'd. from 3.6)

3.2 LPC

In this section we describe the implementation of LPC preprocessor, which computes LPC and LPC derived cepstral coefficients. We also present the experimental results to select the order of the LPC.

3.2.1 LPC Preprocessor

The overall function of the LPC preprocessor is to generate feature vector for every ten milliseconds of speech signal. This feature vectors are later used by vector quantizer for further processing. In the feature vector we generate LPC coefficients ($A_m(t)$), Cepstral coefficients ($C_m(t)$), Weighted cepstral coefficients($\hat{C}_m(t)$) and differential cepstral coefficients($\triangle\hat{C}_m$).

In this subsection different phases of LPC preprocessor (figure 3.8) are briefly explained. The theoretical background for this is available in various references ([14], [3], [13]).

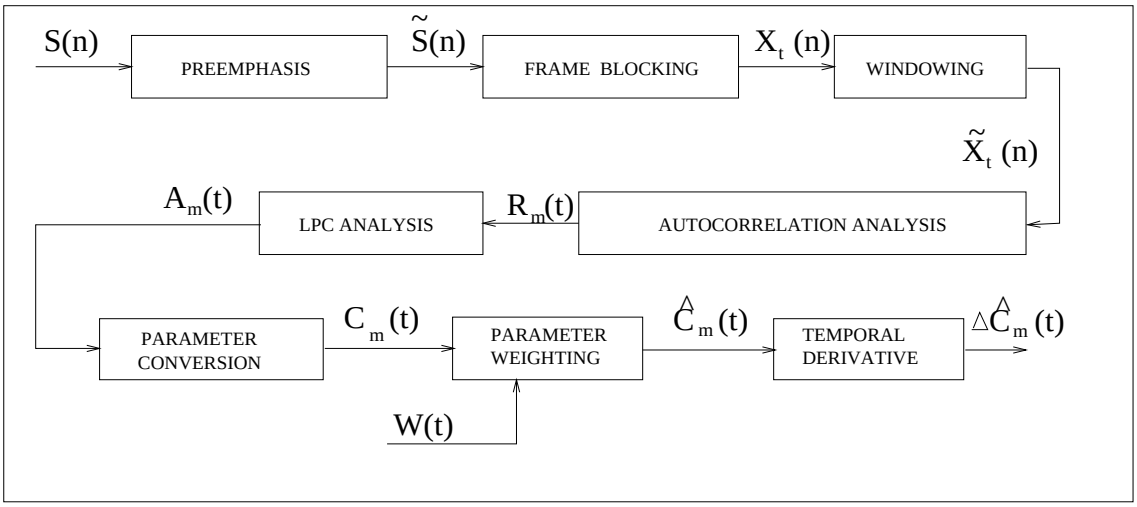


Figure 3.8: LPC preprocessor

PREEMPHASIS : The digitized speech signal $S(n)$, is put through a a first-order FIR filter, to spectrally flatten the signal and to make it less susceptible to finite precision effects later in the signal processing. the computation performed at this stage are,

$$\hat{S}(n) = S(n) - aS(n) \text{ (} a \text{ is taken to be between 0.9 and 1.0)}$$

which refers to a transfer function,

$$H(z) = 1 - az^{-1}$$

FRAME BLOCKING : The preemphasized speech signal is converted into frames of subsequent samples. For this window size is equal to number of samples in $30ms$ of speech. Neighboring frames overlap with number of samples in $20ms$ of speech.

WINDOWING : Each individual frame is windowed to minimize the signal discontinuities at the beginning and the end of each frame. We use the hamming window for this which is transfer function,

$$W(n) = 0.54 - 0.46\cos(2\pi n/(N-1)) \qquad 0 <= n <= N \tag{1}$$

AUTOCORRELATION ANALYSIS : Each frame of the windowed signal is next autocorrelated to get the correlation coefficients using the following formula.

$$R(m) \;=\; \sum_{n=0}^{N-l-m} S(n)S(n+m) \qquad m = 0, 1, \ldots, p$$

LPC ANALYSIS : In this step, we convert each frame of $p+1$ autocorrelations into *LPC parameter set*. The LPC parameter set, which is the set of filter coefficients, can be obtained by solving the set of autocorrelation equations as described earlier.

LPC Parameter Conversion to Cepstral Coefficients : LPC cepstral coefficients set is a very important parameter and can be directly derived from the LPC coefficients. Cepstral coefficients are coefficients of Fourier transform representation of log magnitude spectrum. Cepstral coefficients are most robust and reliable [14] among all the forms of LPC coefficients. Our experiments have also given considerably good performance when cepstral coefficients are used. The cepstral conversion is specified as follows.

$$C_0 = E_0 = R_0 \tag{2}$$

$$C_1 = -A_1 \tag{3}$$

$$C_i = -A_i - \sum_{k=1}^{i-1} \frac{i-k}{i} C_{i-k} A_k \,, \quad 2 \leq i \leq p \tag{4}$$

$$C_i = -\sum_{k=1}^{i-1} \frac{i-k}{i} C_{i-k} A_k \,, \quad i > p \tag{5}$$

Generally, the size of cepstrum coefficient vector, q is chose according to $q \simeq 3p$

Parameter Weighting or Littering : The lower order cepstral coefficients are sensitive to overall spectral slope and the higher order cepstral coefficients are sensitive to the noise. The weighting function $W_t(m)$, essentially deemphasizes the lower and higher order cepstral coefficients i.e around $m = 1$ and $m = q$, by bandpass liftering, i.e, filtering in the cepstral domain. It is described as follows.

$$\hat{C}_m = W_t(m) C_m \,, \quad 1 \leq m \leq q \tag{6}$$

$$W_t(m) = \left[1 + \frac{q}{2} \sin\left(\frac{\pi m}{q}\right) \right] \,, 1 \leq m \leq q \tag{7}$$

Temporal Cepstral derivatives : It has been observed that differential parameters are useful when they are used along with ordinary cepstral coefficients. This is because while the absolute formant locations are sensitive to the speaker variation, the formant slopes are relatively less sensitive to the speaker variation. We therefore compute and use the differential coefficients $\delta\hat{C}_m(t)$ along with $C_m(t)$ in our system.

3.2.2 LPC order selection

In this section we describe how we selected the values for LPC parameters. For this we examined the linear prediction that approximated the real spectrum of the speech and how it changed with the increase of the order of the LPC. In figures 3.9 to 3.15 we show the FFT of a frame in spoken digit *six* of a particular speaker in IITKdigit[1] database. The broken line in each figure shows the LPC predictor spectrum. As we can see in the see figures, the approximation improves with the order of the LPC. The last figure 3.15 is the best approximated one, when LPC order is equal to the number of samples. The LPC smoothens the spectrum with the number of control points determined by the order of the LPC. Visually, it is clear that the LPC

[1] The details of this and other speech databases are given in appendix A.

order *six* or less is not a good approximation. From LPC order *eight* and above, the spectrum approximation seems reasonably good and improving. In our experiments, recognition results are better near LPC order 12 (see tables 3.1 and 3.2). Further it is noticed that the recognition rate does not vary much around the LPC order 12 (see table 3.2). Low order LPC fitted spectrum envelops the long peaks of the actual spectrum without picking the details of short peaks of the spectrum. These long peaks contain the speech information and the short peaks contain the speaker (pitch) information. As the LPC order is increased, the short peaks of the spectrum which are not required for the speech recognition are also picked by the LPC model. This is the reason for decrease in recognition rate at LPC order 18 (see table 3.1). The plots and the final results show that the LPC system can effectively compress the data for spectral information for speech recognition. It could represent spectrum of 216 samples using 12 numbers, which is an order of magnitude smaller.

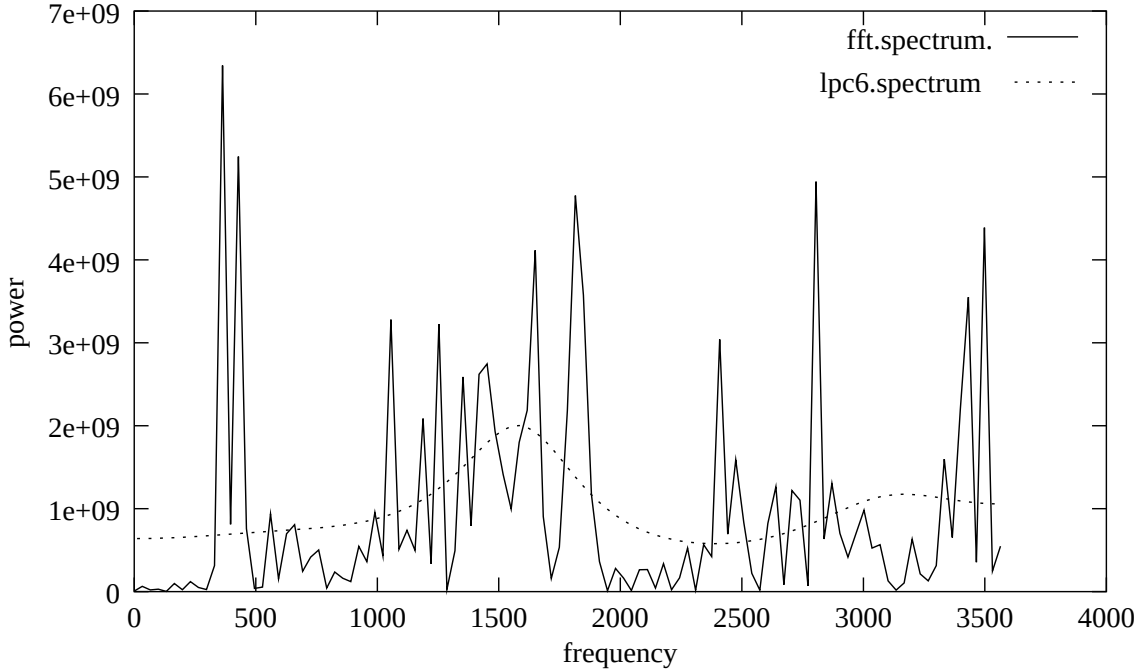


Figure 3.9: Power spectrum of LPC of order 6 plotted against FFT represented power spectrum

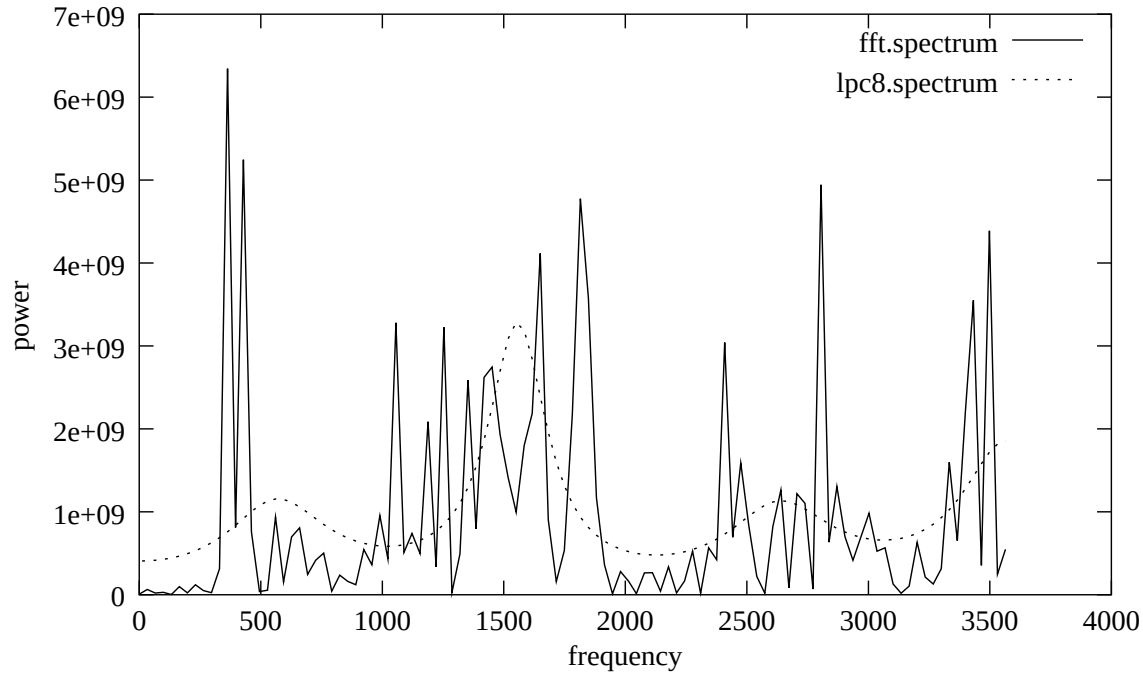


Figure 3.10: Power spectrum of LPC of order 8 plotted against FFT represented power spectrum

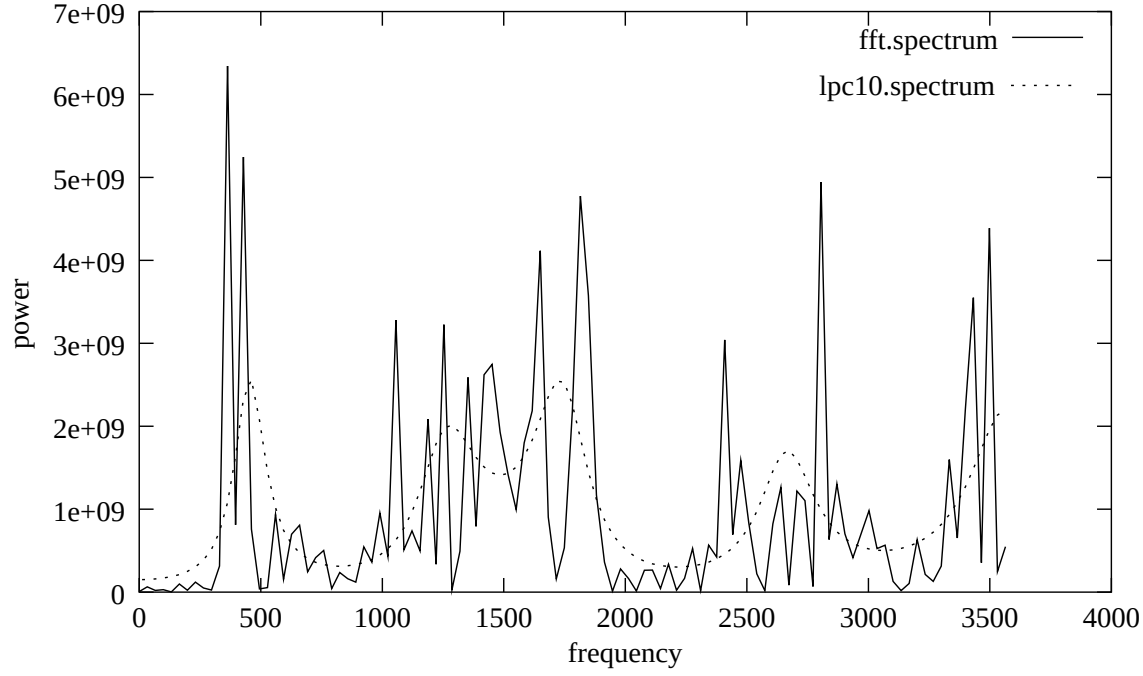


Figure 3.11: Power spectrum of LPC of order 10 plotted against FFT represented power spectrum

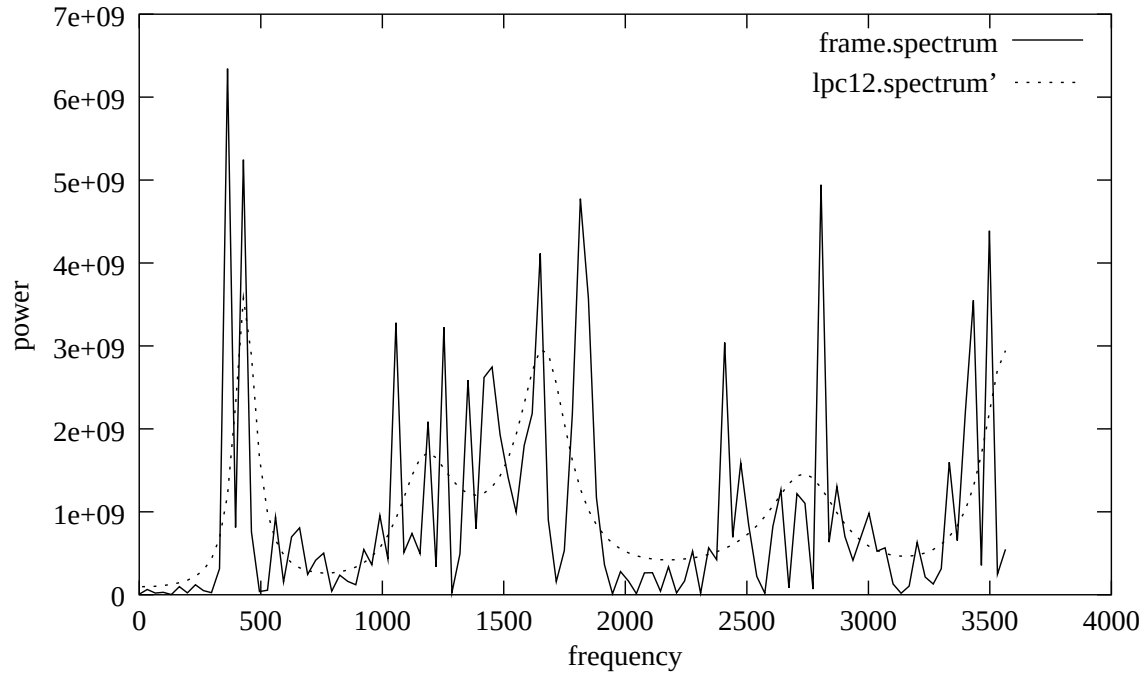


Figure 3.12: Power spectrum of LPC of order 12 plotted against FFT represented power spectrum

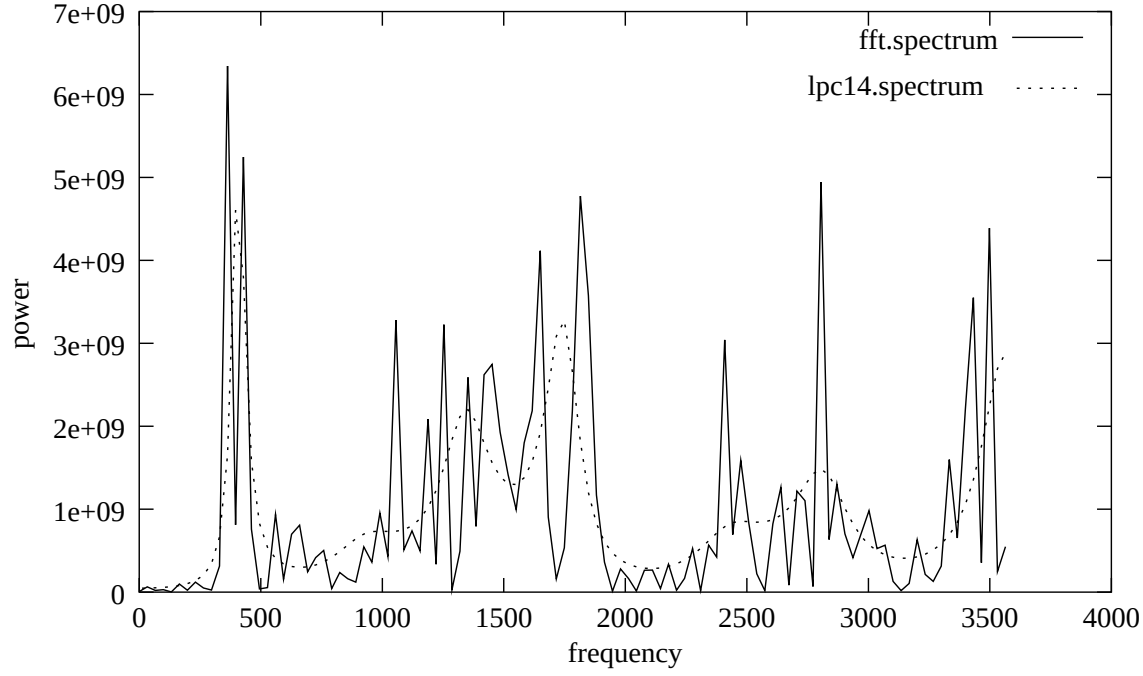


Figure 3.13: Power spectrum of LPC of order 14 plotted against FFT represented power spectrum

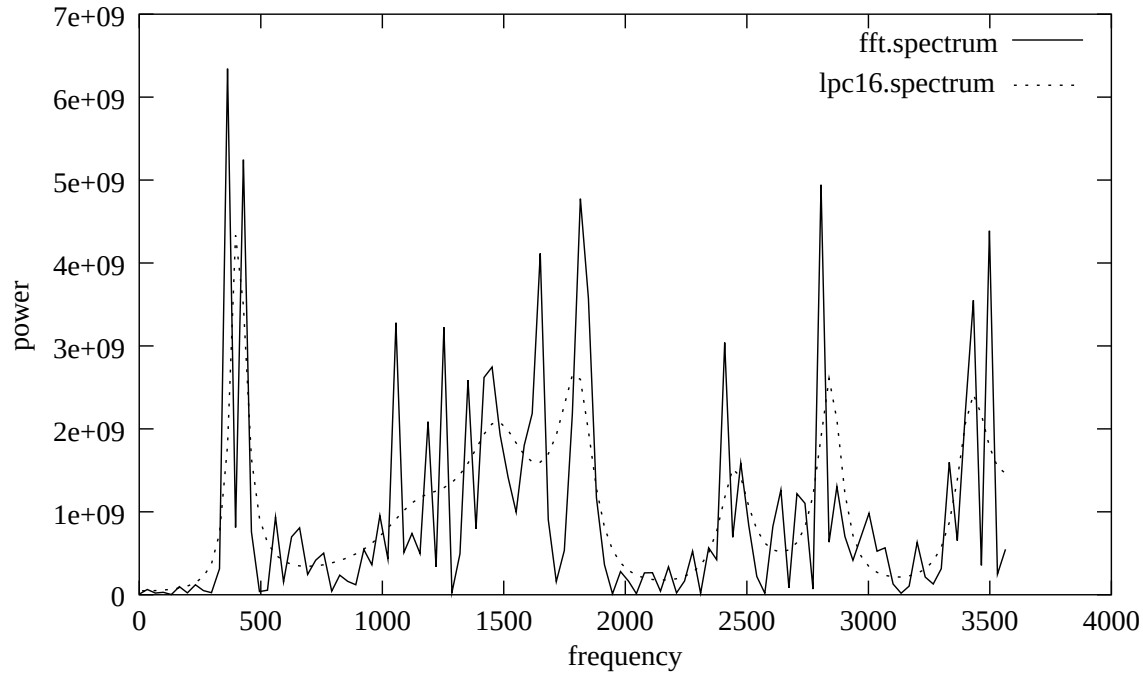


Figure 3.14: Power spectrum of LPC of order 16 plotted against FFT represented power spectrum

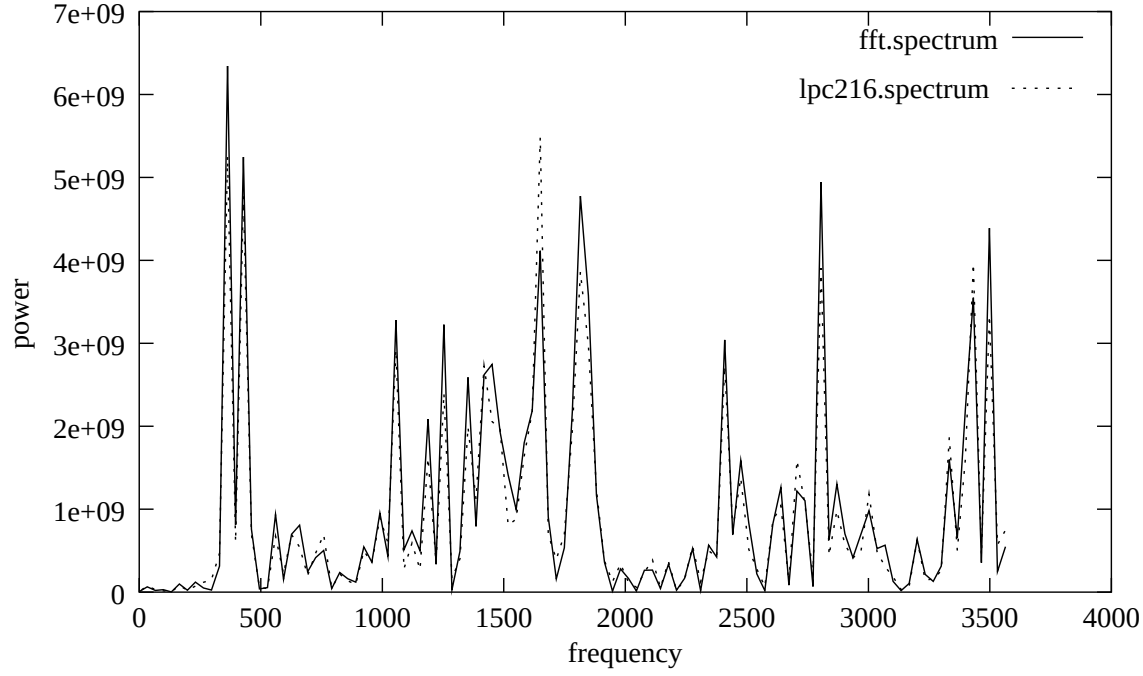


Figure 3.15: Power spectrum of LPC of order 216 plotted against FFT represented power spectrum

LPC order	Percent Recognition
8	88.7
12	91.4
18	89.0

Table 3.1: Initial recognition results with IITKdigitSp73to162 as training database and IITKdigitSp31to72 as testing database with cepstrum code book of size 512

LPC order	Percent Recognition
10	82.96
12	82.5
16	82.9

Table 3.2: Initial recognition results with IITKdigitSp0to72F as training database and IITKdigitSp0to72S as testing database with pure LPC code book of size 64

3.3 Vector Quantization

As discussed earlier, vector quantizer is an important step of the speech recognition. The HMM is built upon the the sequence of symbols which are the code book indices coming from the vector quantizer. Thus even if the feature vector is changed, the HMM implementation remains unchanged.

In this section, we describe in brief the two different vector quantizers, one based on the pure LPC coefficients and another one based on LPC derived cepstral coefficients. The basic mechanism of the vector quantizer has already been described in the chapter 2. Here we describe the distance metric and centroid computation. We used the IITKdigitSp31to72 and IITKdigitSp73to163 databases which generate a total of 83330 training vectors.

3.3.1 VQ Based on Pure LPC coefficients

In this method, the code book is generated using the pure LPC coefficients. The quantizer uses only the spectral shape information in generating the code book.

- *Distance Computation*

We have used likelihood-ratio-distance metric in our implementation of this quantizer. The likelihood-ratio-distance, $d(a_R, a_T)$, between two LPC vectors, a_R and a_T

is defined as,

$$d(a_R, a_T) = \frac{a_R V_T \acute{a}_R}{a_T V_T \acute{a}_T} - 1, \tag{8}$$

where V_T is the autocorrelation matrix of the frame that gave rise to the LPC vector a_T.The autocorrelation matrix V_T is defined as follows [13].

$$\begin{bmatrix} R_0 & R_1 & R_2 & \cdots & R_{p-1} \\ R_1 & R_0 & R_1 & \cdots & R_{p-2} \\ R_2 & R_1 & R_0 & \cdots & R_{p-3} \\ \vdots & \vdots & \vdots & & \vdots \\ R_{p-1} & R_{p-2} & R_{p-3} & \cdots & R_0 \end{bmatrix} \tag{9}$$

Vectors, $\acute{a}_R$ and $\acute{a}_T$ are transpose of the LPC vectors a_R and a_T respectively. The expression $a_T V_T \acute{a}_T$ is the residual error or the energy of the error signal. Since a_T is calculated using the least squares method, minimizing the error residual, the following expressions holds,

$$a_R V_T \acute{a}_R \geq a_T V_T \acute{a}_T \tag{10}$$

and hence,

$$d(a_R, a_T) \geq 0 \tag{11}$$

Updating Centroid

In-order to compute the LPC vector of the centroid of a cell, correlation coefficients for the centroid are calculated as the mean of the respective autocorrelation coefficients of all the members of the cell. Then, the Durbin's method described in reference [13] is used to compute the LPC coefficients.

Quantizer Spectrum Approximation Results

In figures 3.16 to 3.19, we present the spectrum represented by the quantizers of varying sizes. We have used the LPC order of 12 and the same frame that was used in the figures 3.9 to 3.15. The figures show the regenerated spectrum from the code book after vector quantization in broken line.

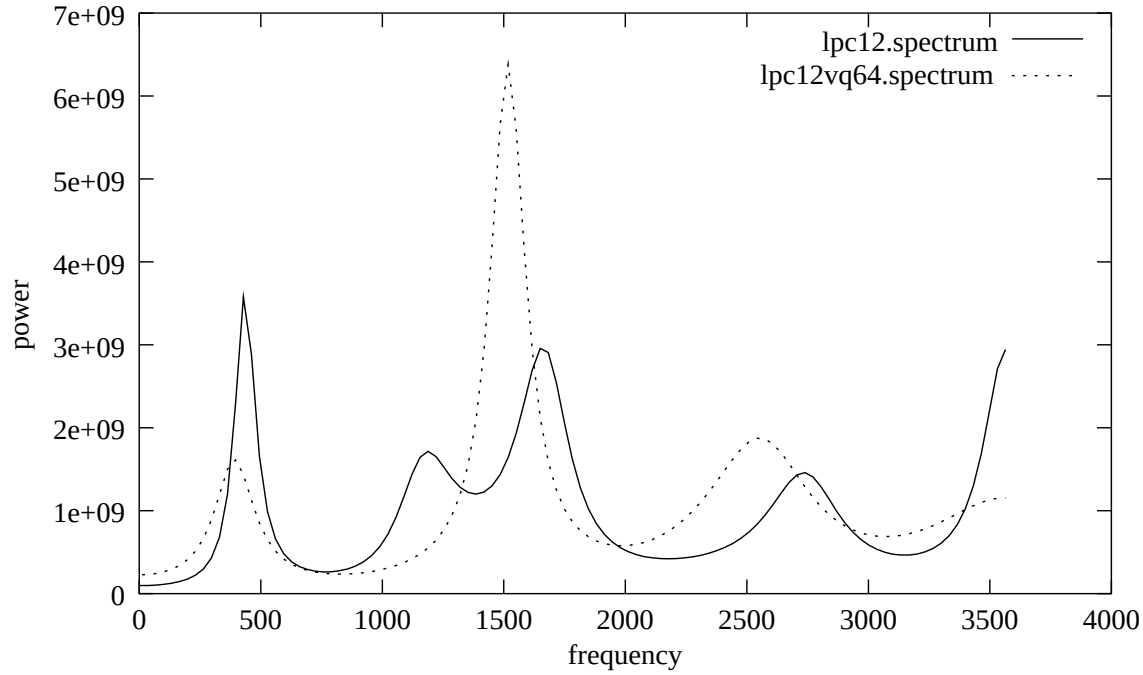


Figure 3.16: The 12^{th} order LPC vector and its quantized versions for code book of size 64

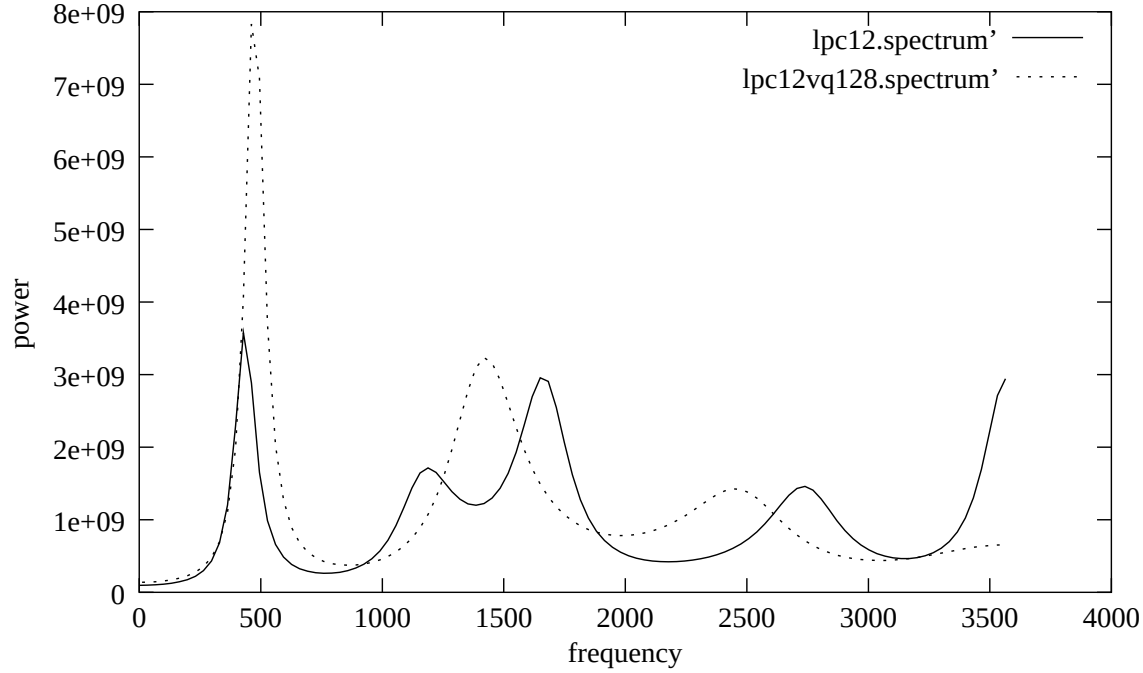


Figure 3.17: The 12^{th} order LPC vector and its quantized versions for code book of size 128

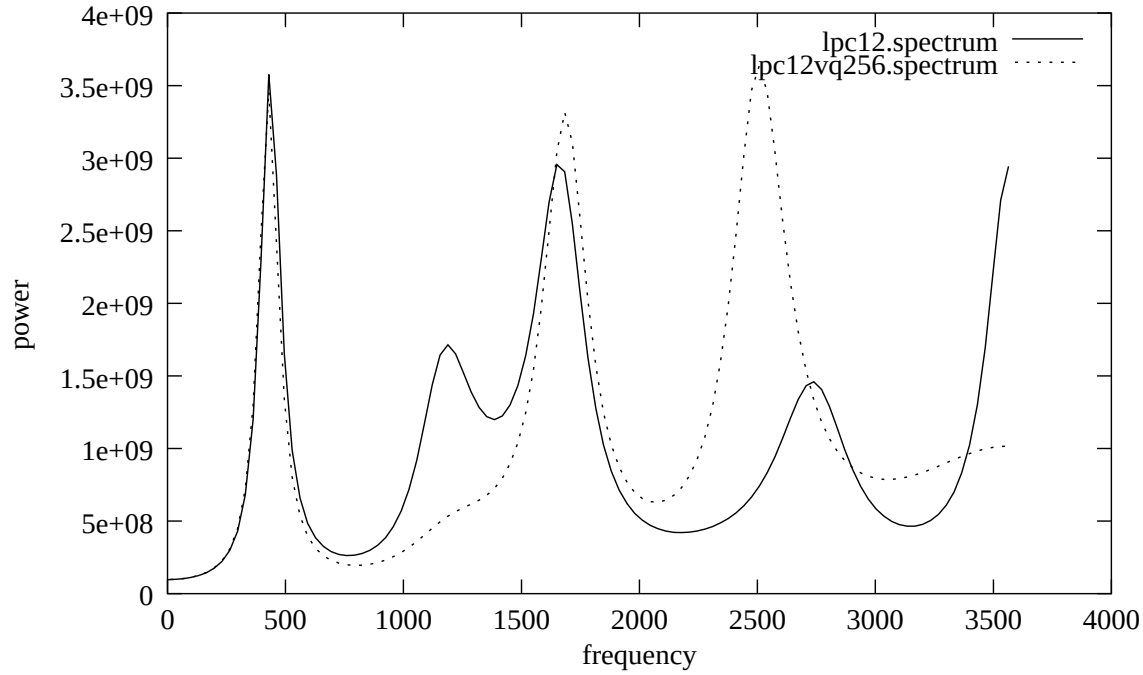


Figure 3.18: The 12^{th} order LPC vector and its quantized versions for code book of size 256

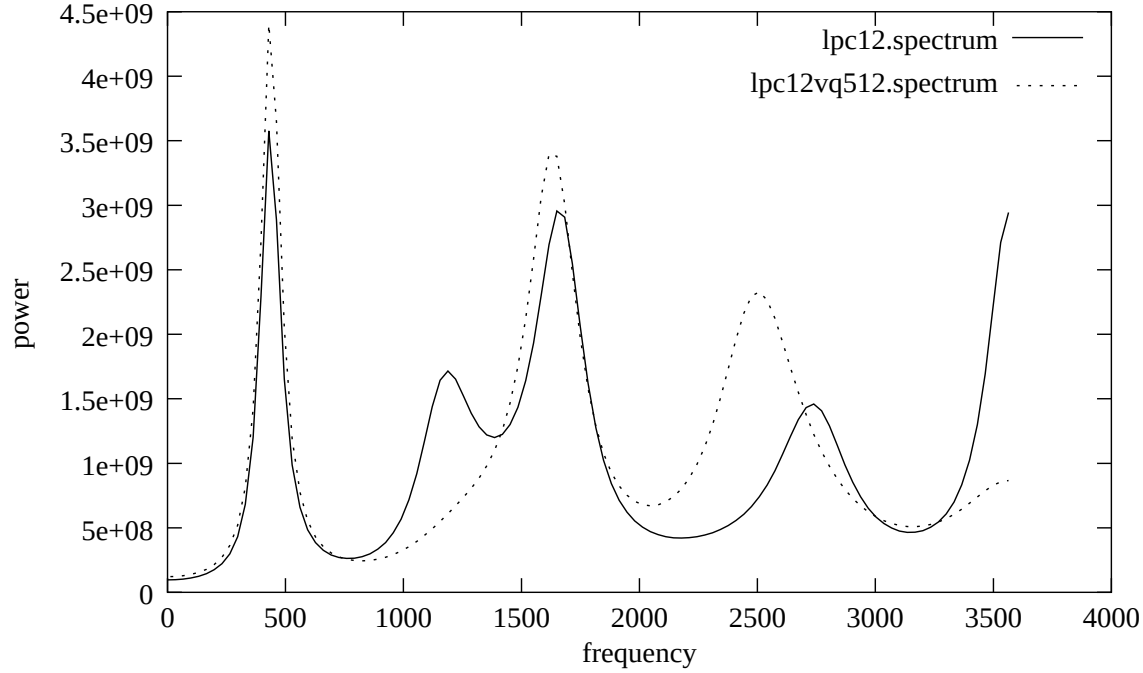


Figure 3.19: The 12^{th} order LPC vector and its quantized versions for code book of size 512

3.3.2 VQ Based on LPC Cepstrum coefficients

Implementation of vector quantizer based on cepstrum coefficients is relatively easy as the distance and centroid computations are simple. The computations remain the same for distance and centroid, whether the cepstral derivatives used or not. However the vector length n is different.

- *Distance Computation*

The distance $d(c,\, \acute{c})$, between two cepstrum vectors $(c_0,\, c_1,\, \cdots, c_n)$ and $(\acute{c}_0,\, \acute{c}_1,\, \cdots,\, \acute{c}_n)$ is calculated as,

$$d(c,\, \acute{c}) \,=\, \sum_{i=0}^{n} (\acute{c}_i - c_i)^2 \tag{12}$$

- *centroid Update*

The centroid computation for a cell containing m cepstral vectors is same as the centroid computation of m points of n-dimensional Eucledion space.

3.3.3 Code book and its size selection

In our work, we chose a vector quantization code book size of 512. The speech recognition based on cepstral coefficients always outperformed the speech recognition based on pure LPC coefficients (see tables 3.3 and 3.4). The code book selection between the pure LPC based one and cepstrum based one is was clearly indicated by the experimental results, as cepstral coefficients based one has been always outperforming the pure LPC based one. Table 3.3 shows the results of such an experiment. while the overall performance of cepstrum based recognizer is satisfactory over the pure LPC based one, the performance for the digit *six* has been very good which was an improvement from 72.1% to 86.0%. So we have decided to go ahead with the cepstrum based vector quantizer. The size of the code book for the cepstrum based vector quantizer has been chosen as 512 based on the experimental results in the table 3.4.

Spoken digit	Number of utterances	Correctly recognized (pure LPC vector of 18 coefficients)	Correctly recognized (cepstrum vector of 18 coefficients)
0	86	86	85
1	86	78	77
2	84	72	78
3	84	74	73
4	73	56	58
5	86	76	75
6	86	62	74
7	85	74	80
8	86	71	72
9	86	77	79
TOTAL	840	726	745
PERCENT		86.4	89.0

Table 3.3: Recognition results with IITKSp73to162 as training database and IITKSp30to72 as test test database to select the quantizer type

Codebook Size	Percent Recognition
128	89.4
256	90.1
512	92.5
1024	92.4

Table 3.4: Recognition results with using cepstral coefficients with varying code book size using IITKdigitSp73to162 as training database and IITKdigitSp30to72 as testing database

3.4 Hidden Markov Model

In this section we discuss the implementation of the Hidden Markov Model.

3.4.1 Implementation Issues

- *HMM Type*

In the experiments conducted by the S. E. Levinson and others [11], the left-right model performed slightly better than the unconstrained model. The left-right model also provides ease of implementation and easy of characterization of the states to the real utterance. We therefore chose the strict left-right model. It has the following additional constraints in the implementation.

1. The first observation is produced while the Markov chain is in a distinguished state, designated as q_1. This implies that for the initial state distribution π, the probability of starting in any other state is zero.
   $\pi = (1, 0, 0, \cdots, 0)$
   and π is not re-estimated.

2. The last observation is produced while the Markov chain is in the distinguished last state designated as q_N. This implies,

$$\beta_T(i) = \begin{cases} 1 & \text{if i=1} \\ 0 & \text{otherwise} \end{cases}$$

   for initialization part of the backward probabilities computation as described in chapter 2.

3. Once the Markov chain leaves a state, that state can not be revisited later. This indicates that each state in the HMM models a small continuous portion of the speech.

3.4.2 Number of Iterations and Stability of Parameters

The transition probability vector and symbol probability vector for each state are determined over a number of iterations. The stability of transition probability vector with number of iterations is given in figure 3.20. Similarly symbol probability vector

convergence is given in figure 3.21 and convergence of log probability of whole training observation sequence is given in figure 3.22. We experimented the speech recognition rates with number of iterations (fig. 3.23) and from that the best recognition rates are obtained around the iteration count being 15 or 50.

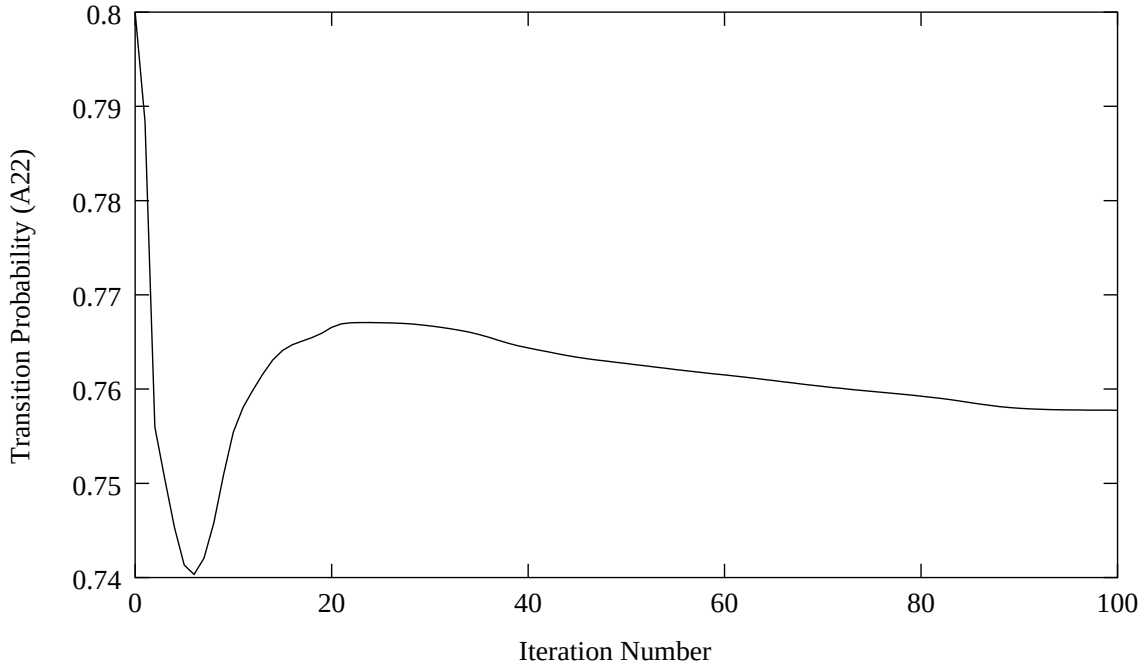


Figure 3.20: Transition probability convergence with number of iterations using TrainDB=OGISp0to79, TestDB=OGISp80to149, LPC order=12, Cepstrum size=16, VQ code book size=512

Number of States in HMM

In our experiments, the recognition rate has been improving with the increase in the number of states of HMM. We have tested up to nine states and the recognition rate was maximum with number of states nine among the experiments conducted. Since, there are other parameters to be tuned for each change in the number of states, we did not experiment with larger numbers of states.

Initial Estimates

We start with initial estimates for the model parameters, iterate reestimating these parameters to optimally train the HMM. There is no theoretical way either to determine

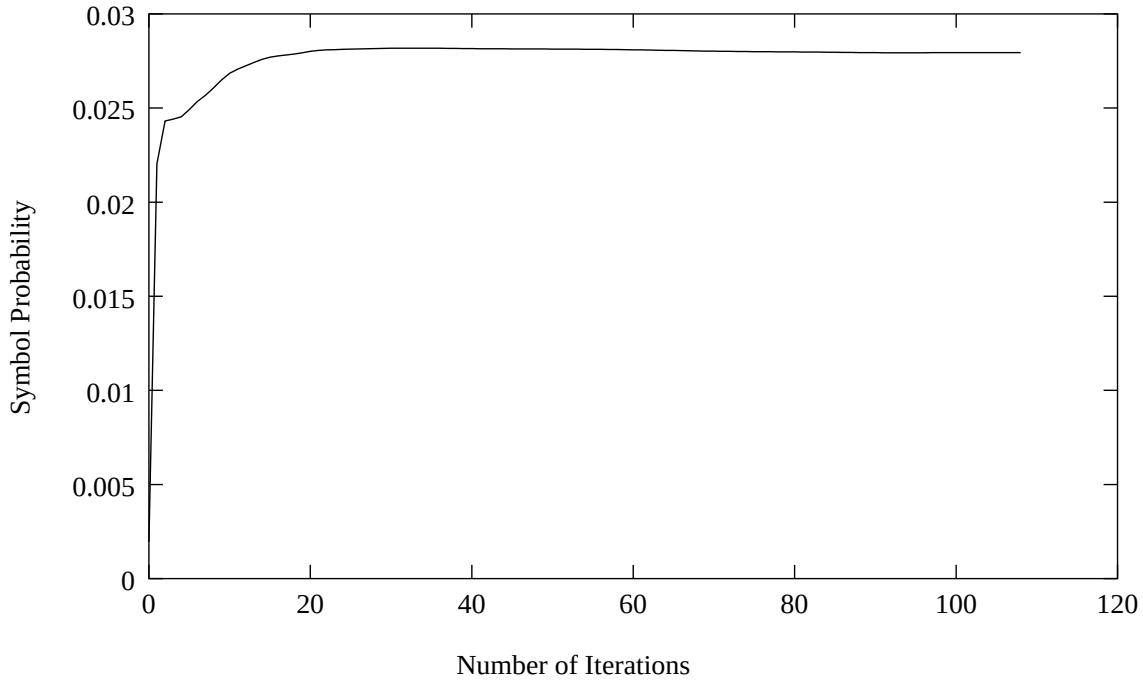


Figure 3.21: Symbol probability convergence with number of iterations using TrainDB=OGISp0to79, TestDB=OGISp80to149,LPC order=12, Cepstrum size=16, VQ code book size=512

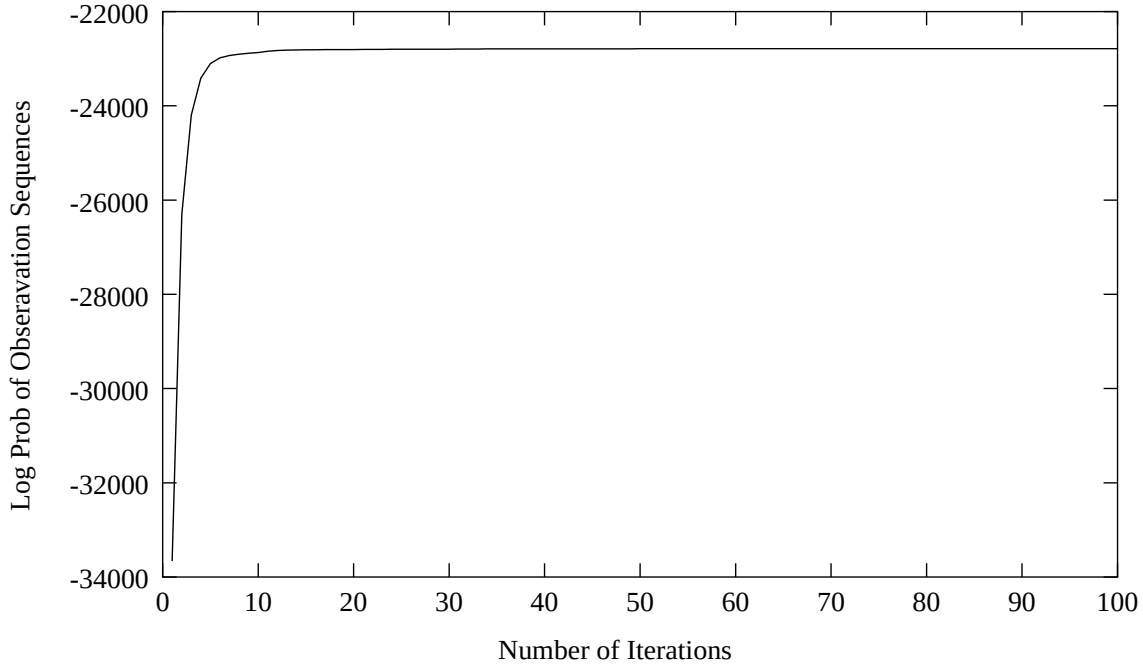


Figure 3.22: $P(O|\lambda)$ convergence with number of iterations;TrainDB=OGISp0to79, TestDB=OGISp80to149,LPC order=12, Cepstrum size=16,VQ code book size=512

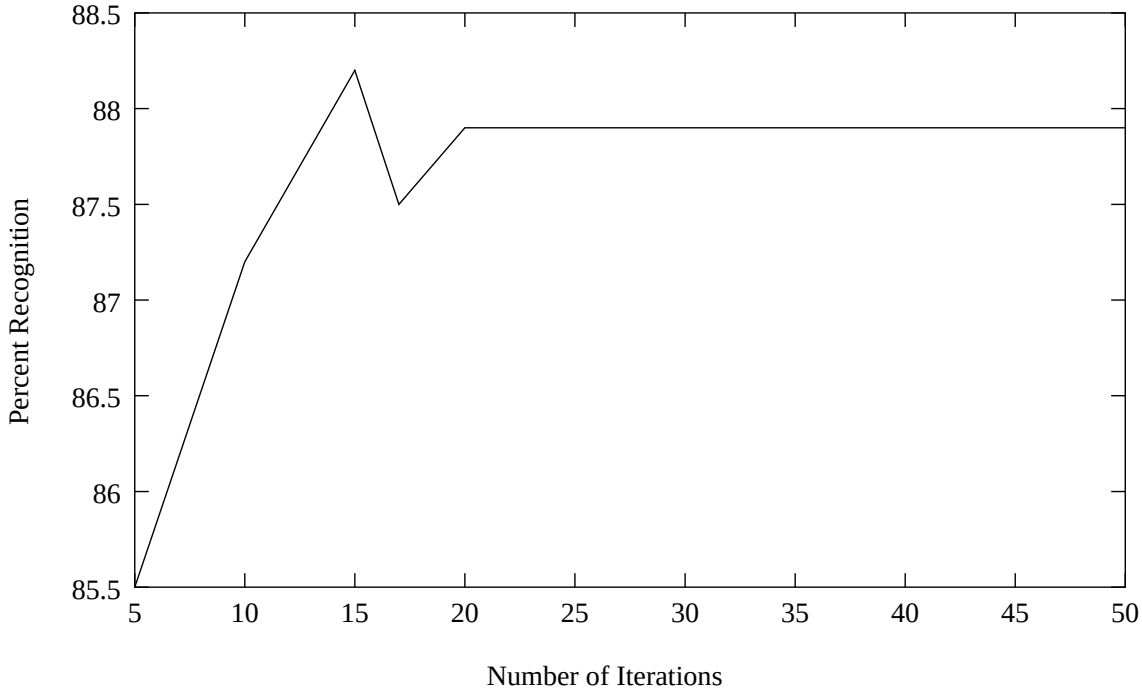


Figure 3.23: Percent recognition with Number of Iterations; TrainDB=OGISp0to129, TestDB=OGI130to149, LPC order=12, Cepstrum size=16, VQ code book size=128

No. of states in HMM	Percent Recognition
6	84.8
7	84.5
8	87.24
9	88.2

Table 3.5: Percent recognition with number of states with TrainDB=OGISp0to79, TestDB=OGISp80to149, LPC order=12, Cepstrum size=16, VQ Code book size=64

or ensure the initial estimates to give the optimum final trained parameters. From the information of the experimental results from the literature [10], [11] and [8], it is evident that any random initial estimates (subjected to our statistical constraints) of state distribution π, and transition probabilities a_{ij} would result in correct parameters after a few iterations. However the same is not true for the symbol probabilities. A good initial estimates for symbol probabilities would give a better performance of the speech recognition. A good estimate is achieved by manually segmenting the observation sequence into states. However a uniform probability distribution for all the observation symbols is good alternative for the initial estimates. Since it is easy to start with uniform probabilities, we have used it as initial estimates of symbol

probabilities. In summary the following are the initial estimates we have used.

- Initial state distribution
  $\pi = (1, 0, 0, \cdots, 0)$
  π is not re-estimated as we have used left-right model.

- Transition Probabilities

$$a_i(j) = \begin{cases} 1 & \text{if i,j = N} \\ 0.8 & \text{if i=j} \\ 0.2 & \text{if j=i+1} \\ 0 & \text{otherwise} \end{cases}$$

- symbol Probability
  $b_{ij} = \frac{1}{M}$, where M is the number of symbols in each state or (i.e. the code book size).

3.4.3 the Basic Speech Recognizer

The basic speech recognition system uses the steps described earlier. In this section we present the results for the basic speech recognition system.

Spoken digit	Number of utterances	Correct percent	0	1	2	3	4	5	6	7	8	9
0	86	100.0	86	0	0	0	0	0	0	0	0	0
1	86	93.02	0	80	0	1	0	2	0	0	0	3
2	84	92.85	3	0	78	0	2	0	0	1	0	0
3	84	96.42	1	0	0	81	0	0	2	0	0	0
4	73	82.19	0	9	1	0	60	0	0	3	0	0
5	86	86.04	0	2	0	0	0	74	0	1	0	9
6	86	84.88	0	0	0	3	1	0	73	0	8	1
7	85	95.29	0	0	0	0	0	0	0	81	0	4
8	86	91.86	0	0	1	0	0	0	3	0	79	3
9	86	97.67	0	0	0	0	0	2	0	0	0	84
TOTAL	842	92.16										

Table 3.6: Results for the basic speech recognizer: TrainDB=IITKSp73to162, TestDB=IITKSp30to72, LPC order =12, cepstrum size =16, code book size=512

The results in the table 3.6 are presented for telephone quality speech. The training was carried out using the speeches of 90 speakers and testing was carried out with

the speeches of 40 different speakers. It was observed that the most of the faulty recognitions are for the digit *six* which is recognized as *eight*, the digit *five* which is confused as digit *nine* and digit *four* which is confused with *one*. Interestingly the converse was not true, that the recognition rate for the digits *one*, *eight* and *nine* are not confused with the digits *four*, *six* and *five* respectively. The recognition rate is 100% for digit *zero* in the tested database.

Similar experiment was done with OGI database which is microphone (connected directly to computer) quality database with laboratory clean recording conditions. As expected, the OGI database has shown higher recognition rate of 96.45% (table 3.7), which is almost 4.0% more than the telephone quality IITK speech database. The experiment was also tried with interchanging the training and testing database (table 3.8) with almost no difference in the recognition rate. The speech recognition rate is 99.84% when same database is used for both training and testing (table 3.9).

Spoken digit	Number of utterances	Correctly recognized	Correct percent
0	65	64	98.46
1	65	65	100.0
2	65	58	89.23
3	65	63	96.92
4	65	64	98.46
5	64	62	96.87
6	65	63	96.92
7	65	62	96.87
8	65	63	96.92
9	65	62	96.87
TOTAL	649	626	96.45

Table 3.7: Results for the basic speech recognizer: TrainDB=OGISp0to84, TestDB=OGISp85to149, LPC order=12, cepstrum size=16, code book size=512

Spoken digit	Number of utterances	Correctly recognized	Correct percent
0	85	84	98.82
1	85	84	98.82
2	85	80	94.11
3	85	85	100.0
4	85	83	97.64
5	85	82	96.47
6	85	84	98.82
7	85	84	98.82
8	85	78	91.76
9	85	83	97.64
TOTAL	850	815	95.88

Table 3.8: Results for the basic speech recognizer: TrainDB=OGISp85to149, TestDB=OGISp0to84, LPC order =12, cepstrum size =16, code book size=512

Spoken digit	Number of utterances	Correctly recognized	Correct percent
0	65	65	100.0
1	65	65	100.0
2	65	65	100.0
3	65	65	100.0
4	65	65	100.0
5	64	64	100.0
6	65	64	98.46
7	65	65	100.0
8	65	65	100.0
9	65	65	100.0
TOTAL	649	648	99.84

Table 3.9: Results for basic speech recognizer: TrainDB=TestDB=OGISp85to149, LPC order =12, cepstrum size =16, code book size=512

Chapter 4

Experiments for Word Rejection and Performance Fine Tuning

With an encouraging result of 92.16% recognition, we have tried to develop heuristics to remove the confusion among various pairs of digits (*eight* and *six*, *four* and *one*, *five* and *nine*). We first discuss our word rejection criterion before describing the other fine-tuning experiments. In this chapter we focus our experiments on the telephone quality speech databases collected at IIT Kanpur.

4.1 Word Rejection

In an online system, speakers may also speak words other than the spoken digits. Such words should be rejected without which they will map on to one of the digits. This problem of word rejection is not trivial. The problem in formulating the word rejection criterion is that we have to use the digit probability in the recognition which itself is a function of number of observation symbols and the duration of the utterance. We should some how normalize probability score to number of observation symbols in the utterance. We approximated normalized probability (NP) in the following way.

$$NP = P^{\frac{1}{N}} \tag{1}$$

Here P is the forward probability score and N is the number of observation symbols in the utterance. We observed that the normalized probability of correctly recognized words is significantly higher than the incorrectly recognized words. The normalized

probability of incorrectly recognized words is significantly less than that of correctly recognized words. We found another interesting phenomenon described as follows. Let $NP0$, $NP1$, $\cdots$, $NP8$ and $NP9$ be the normalized probabilities for a given utterance calculated with the HMMs corresponding to the digits 0, 1, $\cdots$, 8 and 9 respectively. We usually recognize the given utterance as the digit whose corresponding normalized probability is the highest. Let this highest probability be $NPMax1$. Similarly ;et the second highest normal probability be denoted by $NPMax2$. Then we define DNP as the difference between the two as follows,

$$DNP = NPMax1 - NPMax2 \tag{2}$$

It was seen that the DNP for an incorrectly recognized utterance is considerably small compared to a correctly recognized utterance. The figure 4.1 is plot of NP and DNP of the utterances recognized as zero. The 'diamond' is plotted when the utterance is actually ZERO and recognized as ZERO. The 'cross' is plotted when the utterance is not actually ZERO but recognized as ZERO. In the category of 'cross' we included non-digit utterances and other sounds also, which are found in the recording. Figures 4.2 to 4.10 are the similar plots for the digits ONE to NINE. From these figures we can see that most of the incorrectly recognized utterances are located near the origin and x-axis. We therefore imposed additional criterion that the terms NP and DNP should be above some minimum individual thresholds. We have chosen these thresholds differently for different digits (table 4.1. This scheme improves the overall recognition confidence level to 93.5%. The recognition and rejection results are shown in the table 4.2, ignoring the non-digit input. All the data and results presented in this section are based on training database IITKdigitSp73to162 and testing done with the direct speech recording sessions which are used to create the database IITKdigitSp31to72. Considering only the digit utterances, overall recognition rate is 90.7%, in which 83.2% utterances are accepted and remaining 7.5% are rejected. Among the 9.3% wrongly recognized utterances, 4.1% are are accepted and 5.2% are rejected. It means when a digit is accepted, the confidence level is 95.3%, which is a significant improvement against 90.7% when rejection criterion is not used.

4.2 Performance Fine-tuning

Our experiments to improve the performance of the recognition can be classified into two categories. In one category of experiments we tried to improve the performance

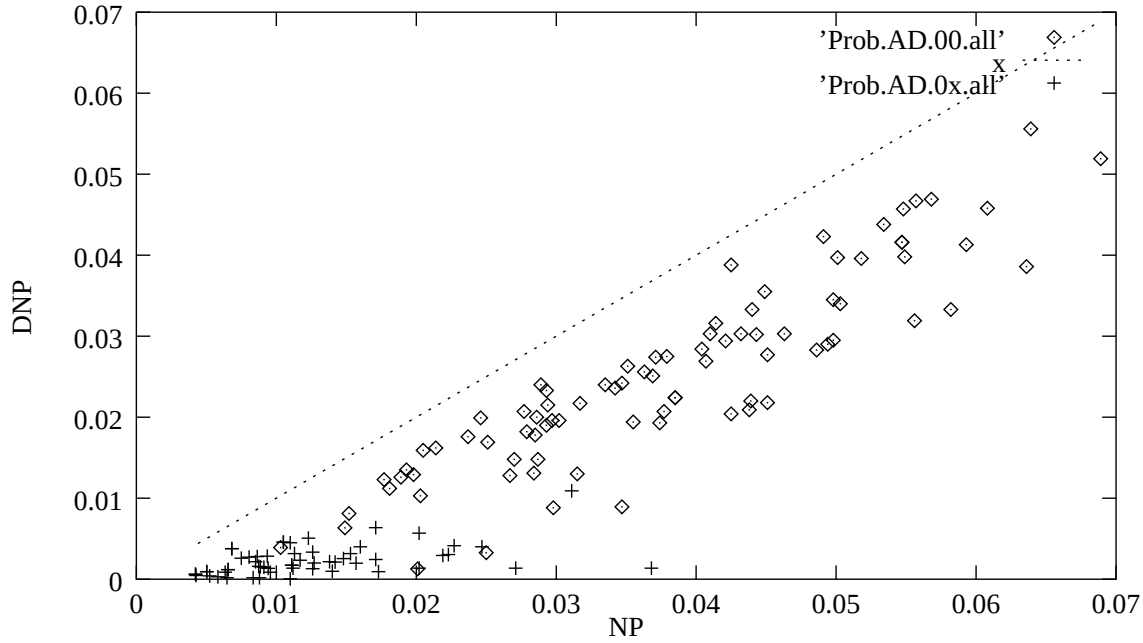


Figure 4.1: NP vs DNP plot when the recognition output is ZERO

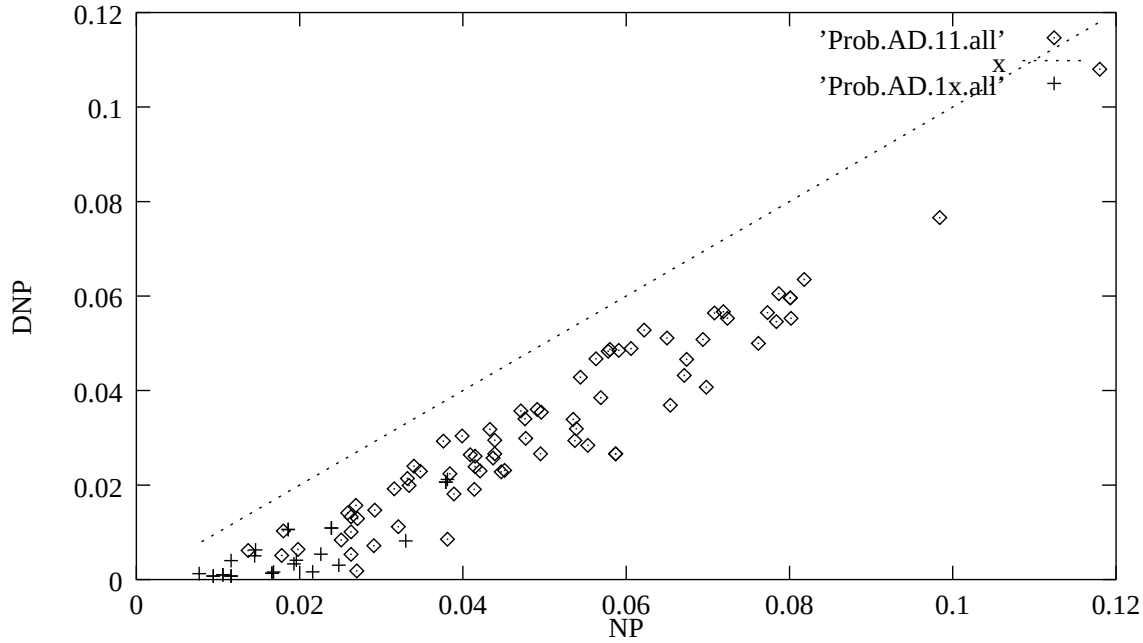


Figure 4.2: NP vs DNP plot when the recognition output is ONE

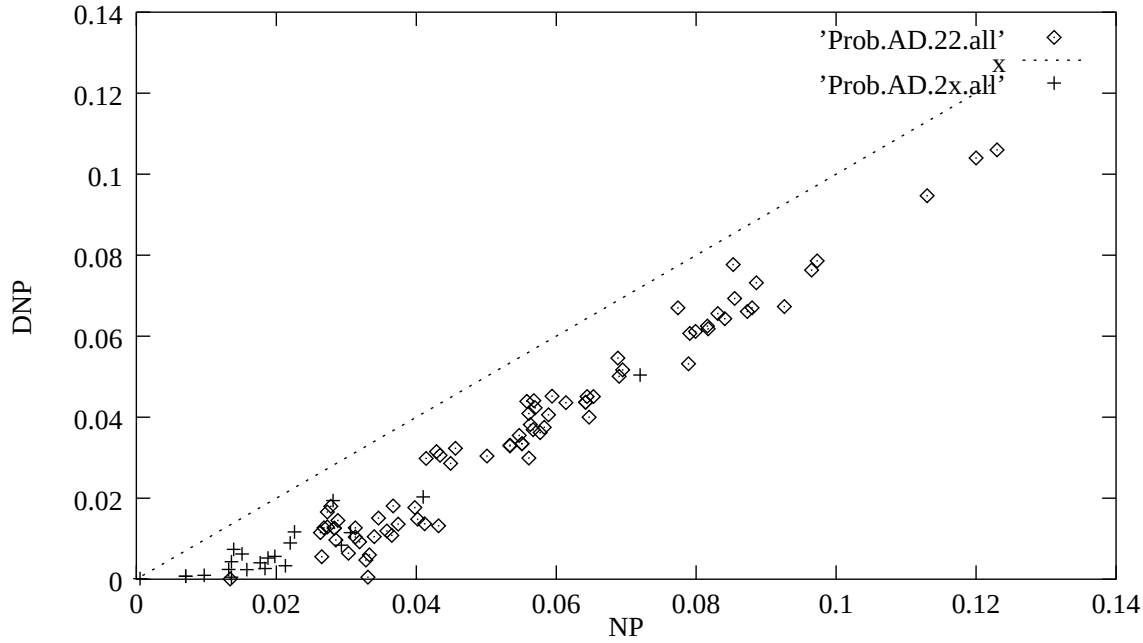


Figure 4.3: NP vs DNP plot when the recognition output is TWO

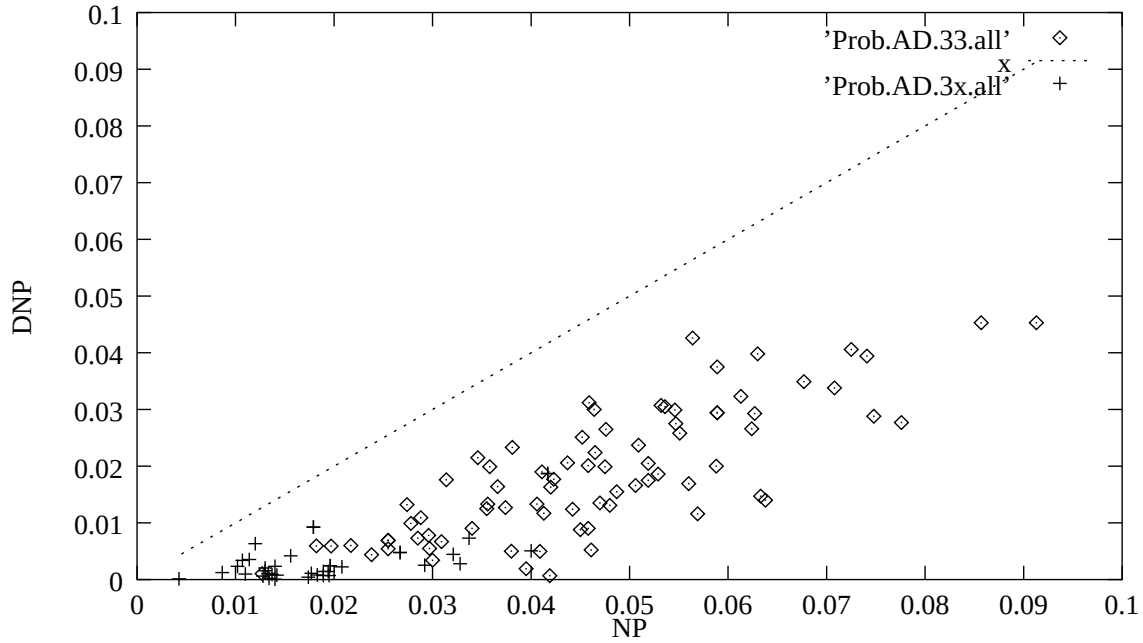


Figure 4.4: NP vs DNP plot when the recognition output is THREE

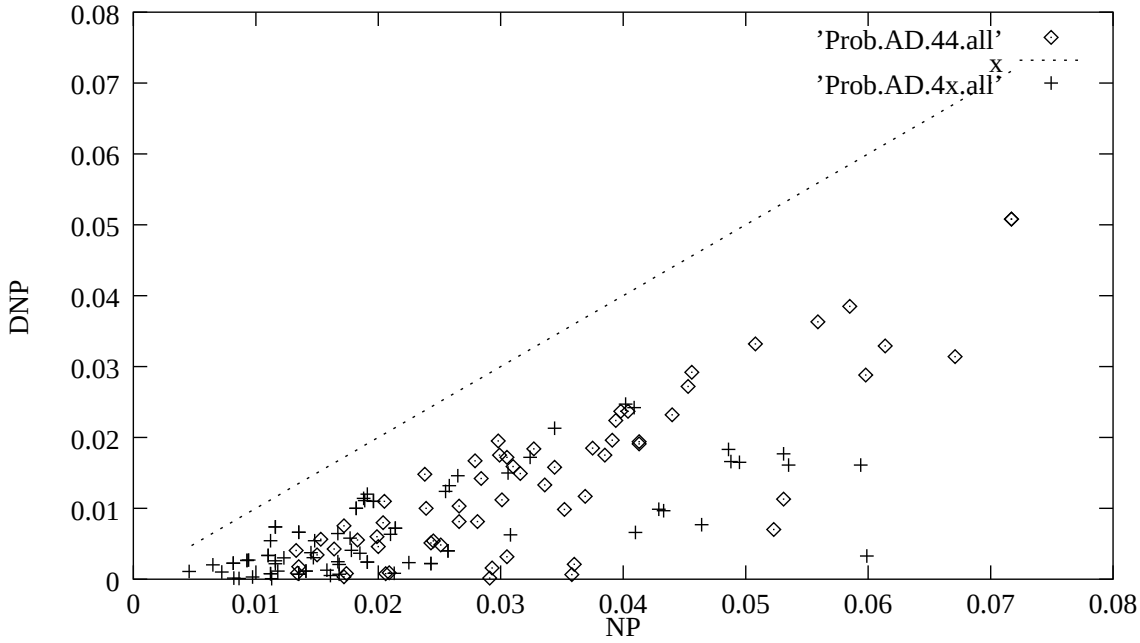


Figure 4.5: NP vs DNP plot when the recognition output is FOUR

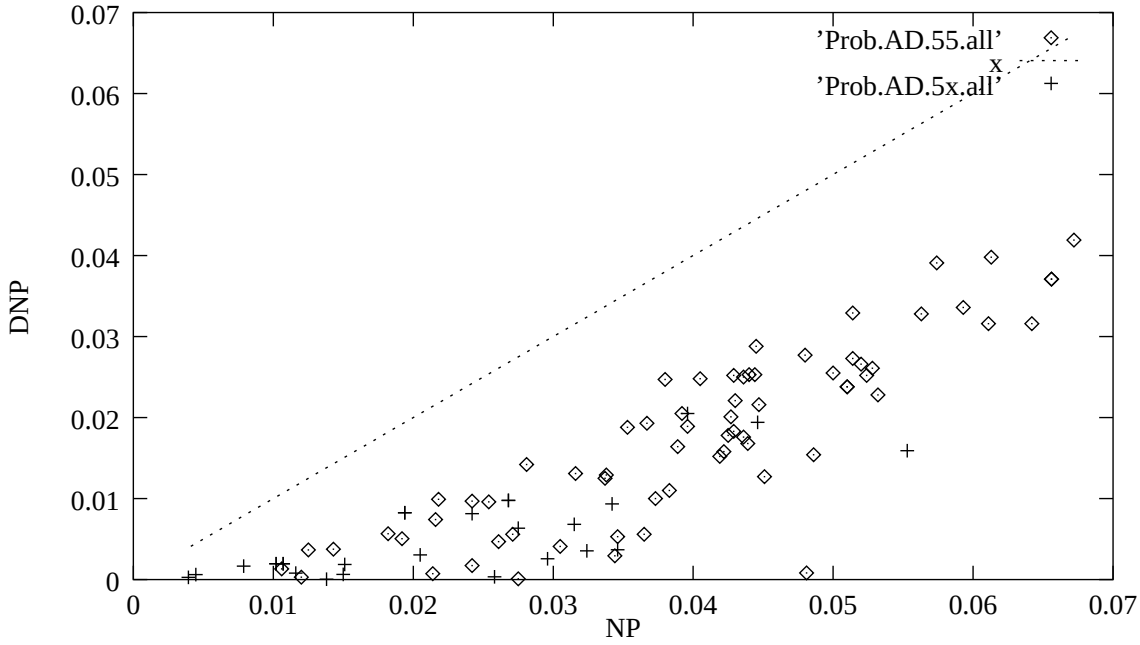


Figure 4.6: NP vs DNP plot when the recognition output is FIVE

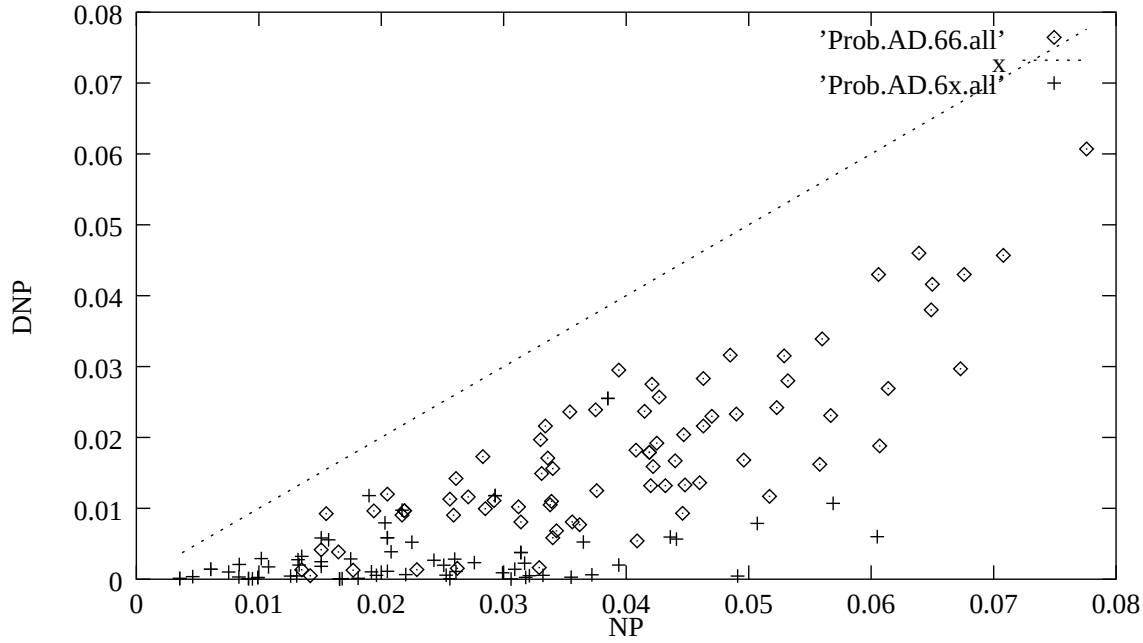


Figure 4.7: NP vs DNP plot when the recognition output is SIX

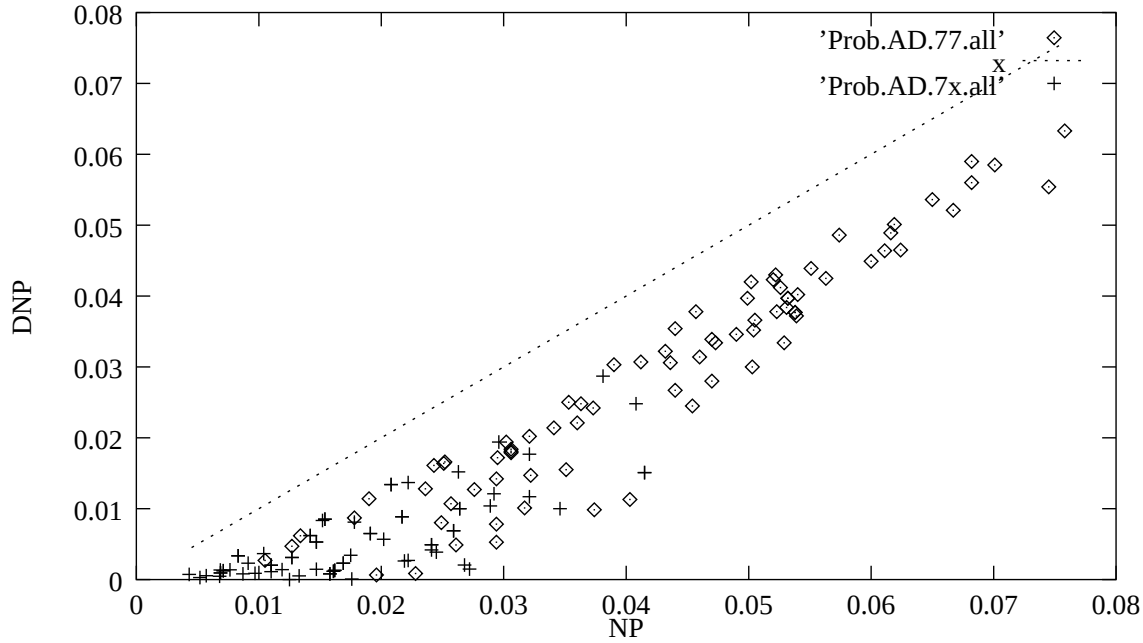


Figure 4.8: NP vs DNP plot when the recognition output is SEVEN

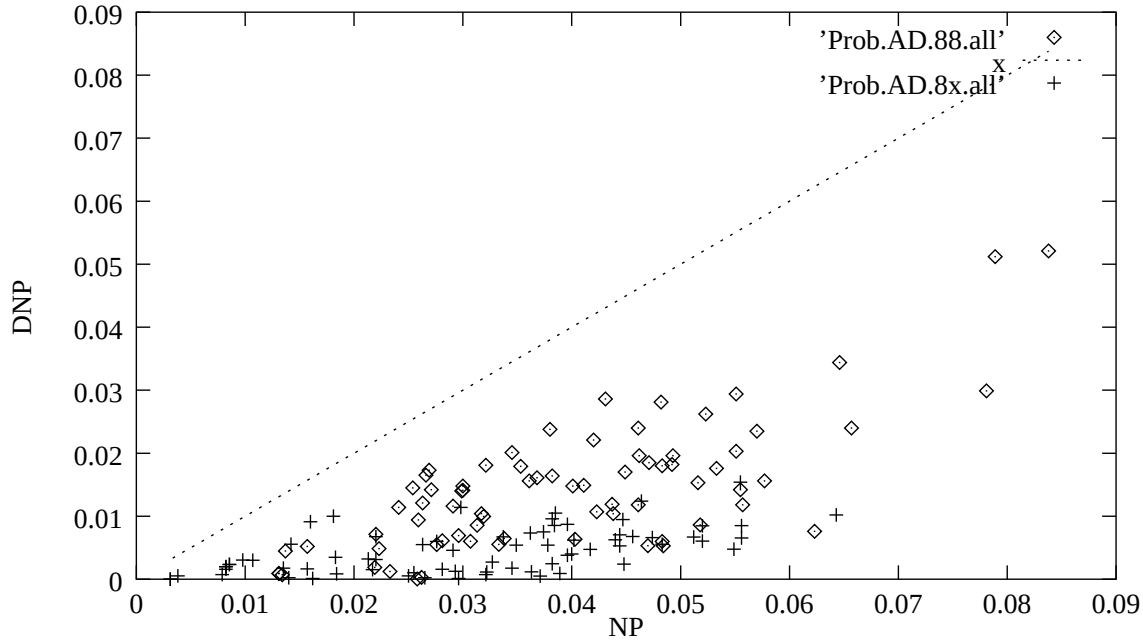


Figure 4.9: NP vs DNP plot when the recognition output is EIGHT

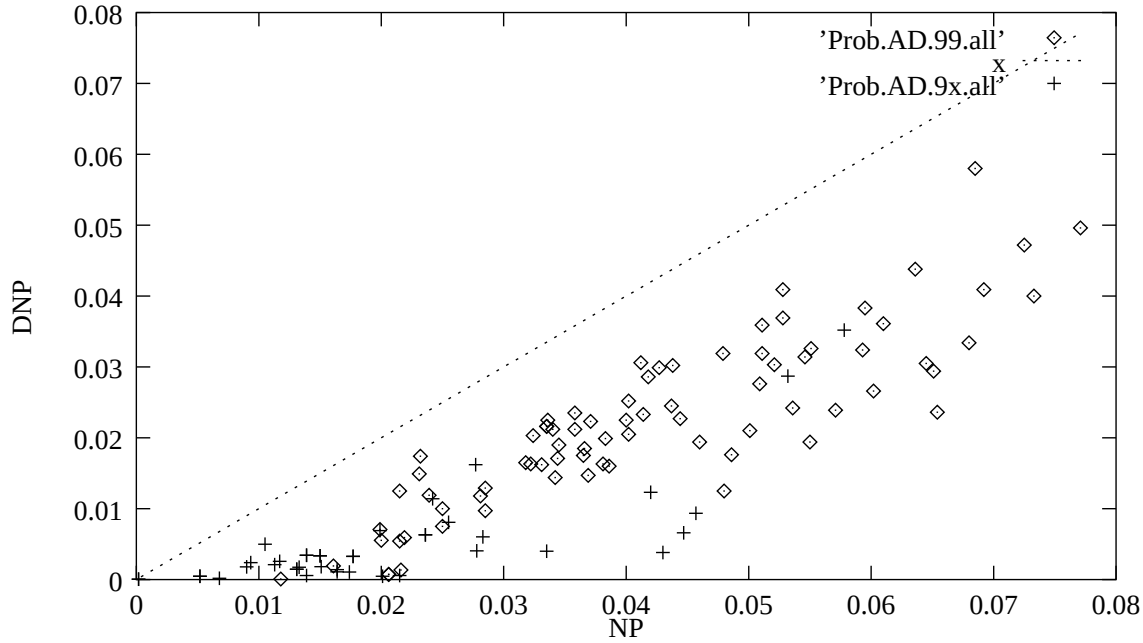


Figure 4.10: NP vs DNP plot when the recognition output is NINE

Digit	NP Threshold	DNP Threshold
0	0.015	0.006
1	0.017	0.004
2	0.025	0.002
3	0.017	0.004
4	0.0145	0.003
5	0.012	0.001
6	0.0155	0.004
7	0.017	0.004
8	0.02	0.005
9	0.02	0.005

Table 4.1: NP and DNP thresholds used with IITKdigitSp73to162 as training database and Testing with direct recording

Digit Digit	Correctly Accepted	Wrongly Accepted	Wrongly Rejected	Correctly Rejected
0	81	0	5	0
1	74	7	3	2
2	75	4	4	1
3	70	3	6	5
4	58	3	7	6
5	65	6	6	6
6	73	2	6	3
7	70	0	5	8
8	56	5	13	8
9	66	4	7	4
TOTAL	688	34	62	43
PERCENT	83.2	4.1	7.5	5.2
PERCENT (Within Accepted)	95.3	4.7		

Table 4.2: Recognition and rejection results for digit inputs with IITKdigitSp73to162 as training database and Testing with direct recording

by focusing on the signal processing front end The other category of experiments were focussed on HMM back end.

4.2.1 Experiments with Signal Processing Front end

In these experiments we introduced a few new features in the feature vector as follows.

- *Frame Energy*

The energy of the frame is appropriately scaled and added to the feature vector. In table 4.3, column III, the recognition results are given with this additional parameter.

Spoken digit	Number of utterances	recognition with energy	recognition with duration	recognition with relative position	with duration and rel.pos.
0	86	86	85	85	86
1	86	76	78	79	81
2	84	76	77	80	76
3	84	78	77	79	80
4	73	67	63	62	62
5	86	75	76	79	77
6	86	78	69	75	74
7	85	81	81	82	79
8	86	76	78	74	79
9	86	83	83	83	85
TOTAL	842	776	767	778	779
PERCENT		92.16	90.09	92.39	92.51

Table 4.3: Recognition results of basic recognizer with additional features using TrainDB=IITKSp73to162, TestDB=IITKSp30to72

- *Utterance Duration*

The duration of the utterance was added in the the feature vectors of all the frames of that utterance. In table 4.3, column IV, the recognition results are given with this additional parameter.

- *Frame Relative Position*

The relative position of the frame with in the utterance is added to the feature vector. In table 4.3, column V, the recognition results are given with this additional parameter.

- *Combinations*

Various combinations of these three features were used in getting the recognition rates. We got the best recognition rate of 92.51% when utterance duration and relative frame position were combined as shown in table 4.3, column VI.

4.2.2 Experiments with trained HMMs

In this section we present the experiments and results for improving the performance of trained HMMs. In all these experiments, we had incorporated the extension of features set by incorporating the duration of utterance and relative position of the frame. We present here only those experiments which gave us the best recognition results. We first define three basic operations that we used in these experiments.

1. *HMM state Tie:* In this operation, given state $s1$ in some trained HMM and state $s2$ in other trained HMM, we manually force the symbol probability distribution in these states to be identical. To understand its usefulness, consider the HMM models trained for digits FIVE and NINE. These two spoken digits have common phoneme /ai/. Let us assume that the states corresponding to the phone /ai/ are not trained well enough for the HMM model of digit FIVE, whereas the corresponding states in the HMM model for digit NINE are trained very well. Now it is possible that many of the spoken digits 'FIVE' are recognized as 'NINE' because of the better performance of states corresponding to /ai/ in its HMM model. This situation can be handled better by forcing the respective pair of states in the two HMMs to have the same symbol probability distribution. Let us say states s_i and s_{i+1} correspond to phoneme /ai/ in HMM trained for digit 'FIVE' and states s_j and s_{j+1} correspond to phoneme /ai/ in HMM trained for digit 'NINE'. We can tie the states s_i and s_j together. Similarly the states s_{i+1} and s_{j+1} are tied together. Let us represent the above tie operation $HMMTye((d1, s1), (d2, s2))$.

2. *HMM embedded scaling:* It is some times desirable to either emphasize or deemphasize certain portions of the utterance while calculating the observation sequence probability. Following are some of the cases where it can be useful.

    - If the HMM tying results in poor recognition rate for the two digits involved in the tying, as many of these two digits are recognized as some other third

digit. In such a case we may wish to emphasize the symbol probabilities of the states involved in the HMM tying.

- If the HMM tying results in poor recognition rate for some digit other than the two digits involved in the tying, as many of the utterances of the third digit are recognized as one of the two digits involved in the tying. In such a case we may wish to de-emphasize the symbol probabilities of the states involved in the HMM tying.
- In case of confusion between digits 'FIVE' and 'NINE', instead of tying states corresponding to /ai/, we may deemphasize these states symbol probabilities and/or emphasize the state symbol probabilities of other states.

Let us denote the above scaling operation as $HMMEmbedScale(d, s, f)$.

3. *Minimum Probability Criterion* Given a HMM trained for the digit d and state s, this operation ensures each and every symbol probability in that state to be greater than or equal to ϵ. Let us denote this operation as $MinB(d, s)$.

In the following discussion, we represent these operations as follows.

- $HMMTye((d1, s1), (d2, s2))$**:** To tie state s_1 of HMM of digit d_1 to state s_2 of HMM of digit d_2
- $HMMEmbedScale(d, s, f)$**:** To scale the observation probability of each symbol in state s by a factor of f.
- $MinB(d, s)$**:** To put the minimum limit on the symbol probabilities in state s of digit d.

1. *Experiment 1:* State 6 of digit 9 is deemphasized and limit is put on the symbol probabilities as follows.
   (a) $HMMEmbedScale(9, 6, 0.01)$
   (b) $MinB(9, 6)$

   This step improved overall recognition rate from 92.51% to 92.63% (table 4.4).
2. *Experiment 2:* State 6 of digits 0 and 6 are tied together and then deemphasized. These are then put through the minimum limit as follows.

(a) $HMMTye((0,6),(6,6))$

(b) $HMMEmbedScale(0,6,0.01)$

(c) $HMMEmbedScale(6,6,0.01)$

(d) $MinB(0,6)$

(e) $MinB(6,6)$

This step improved overall recognition rate from 92.63% to 93.35% (table 4.5).

3. *Experiment 3:* In order to reduce confusion between digits 5 and 9, state 5 of these two digits are tied and passed through minimum limit as follows.

   (a) $HMMTye((5,5),(9,5))$

   (b) $MinB(5,5)$

   (c) $MinB(9,5)$

   This step improved recognition rate from 93.35% to 93.47% (table 4.6). As a side effect, the recognition rate of digit7 was also improved.

4. *Experiment 4:* The digit 4 was confused with other utterances. States 4, 7 and 9 of digit 4 were emphasized as follows.

   (a) $HMMEmbedScale(4,6,2)$

   (b) $HMMEmbedScale(4,7,2)$

   (c) $HMMEmbedScale(4,9,2)$

   This step improved recognition rate from 93.47% to 93.94% (table 4.7)

5. *Experiment 5:* States 6,7 and 9 of digit 2 are emphasized.

   (a) $HMMEmbedScale(2,6,2)$

   (b) $HMMEmbedScale(2,7,2)$

   (c) $HMMEmbedScale(2,9,2)$

   This experiment improved recognition rate from 93.94% to 94.3% (table 4.8).

Spoken digit	Number of utterances	Correctly recognized
0	86	86
1	86	81
2	84	76
3	84	80
4	73	62
5	86	78
6	86	74
7	85	79
8	86	79
9	86	85
TOTAL	842	780
PERCENT		92.63

Table 4.4: Recognition results after experiment 1, using TrainDB=IITKSp73to162, TestDB=IITKSp30to72

Spoken digit	Number of utterances	Correctly recognized
0	86	86
1	86	81
2	84	76
3	84	84
4	73	62
5	86	77
6	86	74
7	85	79
8	86	82
9	86	85
TOTAL	842	786
PERCENT		93.35

Table 4.5: Recognition results after experiment 2, using TrainDB=IITKSp73to162, TestDB=IITKSp30to72

Spoken digit	Number of utterances	Correctly recognized
0	86	86
1	86	81
2	84	76
3	84	84
4	73	62
5	86	76
6	86	74
7	85	81
8	86	82
9	86	85
TOTAL	842	786
PERCENT		93.47

Table 4.6: Recognition results after experiment 3, using TrainDB=IITKSp73to162, TestDB=IITKSp30to72

Spoken digit	Number of utterances	Correctly recognized
0	86	86
1	86	81
2	84	72
3	84	84
4	73	70
5	86	76
6	86	74
7	85	81
8	86	82
9	86	85
TOTAL	842	791
PERCENT		93.94

Table 4.7: Recognition results after experiment 4, using TrainDB=IITKSp73to162, TestDB=IITKSp30to72

Spoken digit	Number of utterances	Correctly recognized
0	86	85
1	86	81
2	84	78
3	84	83
4	73	69
5	86	76
6	86	74
7	85	81
8	86	82
9	86	85
TOTAL	842	794
PERCENT		94.3

Table 4.8: Recognition results after experiment 5, using TrainDB=IITKSp73to162, TestDB=IITKSp30to72

Chapter 5

An Application: Interactive Voice Response System for Enquiring JEE Application Status

In this chapter we discuss the design and implementation of an *interactive voice response system (IVR)* application, which we developed for answering the queries regarding the JEE application status. We first discuss the dialogue design and then its implementation using the technology developed.

5.1 Dialog Design

Here we present the dialog designed for a single interactive session between the IVR and the user. The user dials the specified number to the IVR through the modem. The following the dialog design for the IVR.

- *IVR* : Welcome to the IIT Kanpur IVR for answering the queries for your JEE application status. Please speak the digits in the application number slowly one at a time after the beep.
- *IVR* : Plays the beep
- *Caller* : Speaks the individual digits of application number.
- *IVR* : Your application number ***** has reached the JEE office.

5.2 Implementation

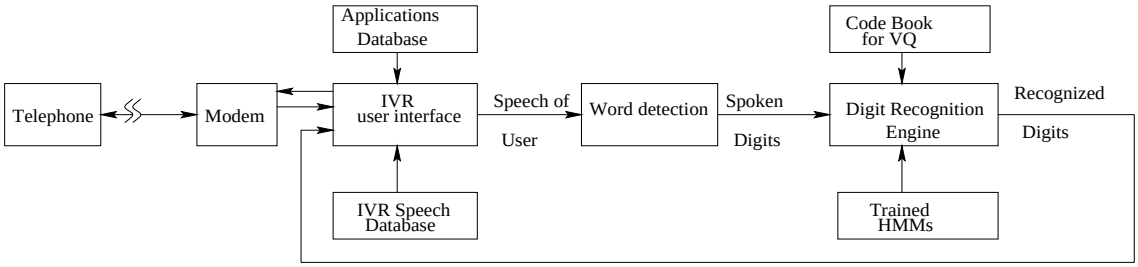


Figure 5.1: Interactive Voice Response System for JEE Application Status Enquiry

The block diagram of the interactive voice response system that we implemented is shown in the figure 5.1. The user interface part of the IVR monitors modem and when there is an incoming call from the user, it picks up the phone and plays the pre-recorded welcome message and then generates a beep. The IVR then records the voice of the speaker for ten seconds in a buffer. The recorded voice is then passed to the word-detection module. Word-detection module extracts the isolated spoken digits and passes them to the digit recognition engine. Digit recognition engine accepts one spoken digit at a time and recognizes the digit in the spoken speech. The recognized digit is then passed to the IVR user interface module. IVR user interface module assembles the digits into number and searches it in the database of received applications numbers. An appropriate message is played back. For digits, pre-recorded messages are played. Finally the call is disconnected and the entire process is repeated for another call.

Chapter 6

Conclusions and future work

In this thesis we implemented an isolated speaker independent spoken digit recognition system for telephone quality speech. In particular we have been able to achieve the following.

1. *Speech Database:* We built spoken digit database of 163 speakers.
2. *Speech Recognizer:* We implemented every part of the speech recognizer. we were able to formulate word rejection criterion. We could improve the speech recognition rate by fine-tuning different parts of the basic speech recognizer.

Based on our experiments, we can conclude the following.

6.1 Conclusions

- The heuristics used in word detection improved the word detection rate and speech recognition rate.
- Cepstral coefficients are better than the LPC coefficients.
- Addition of differential features improved the performance.
- HMM tye and embedded weight heuristics improved the recognition rate.
- Our word rejection criterion is valid and improved the confidence of the recognized digit.

- Relatively clean telephone quality speech has better recognition rate than the relatively noisy telephone quality speech.

6.2 Limitations

- The IITKdigit speech database has very few female speakers and the recognizer built using this database as training database may perform poorly for female speakers.
- The energy thresholds used in the word detection algorithm are specific to our modem and may need to be changed if the modem is replaced.

6.3 Future Work

- Speech of the female speakers can be added to the IITKdigit database
- Word detection algorithm can be modified to dynamically adopt the energy thresholds.
- The recognition rate can be studied, how it is effected with HMM states more than nine.
- The recognizer can be enhanced to recognize the continuous speech.
- Initiative can be taken to start building IITK speech tool kit and IITK speech database for future research.

Bibliography

[1] Baker, J. K. The Dragon System - An Overview. *IEEE Transactions on Acoustics, Speech, Signal Processing ASSP-23*, 1 (February 1975), 24–29.

[2] de Souza, P. V. Statistical Tests and Distance Measures for LPC Coefficients. *IEE Transactions on Acoustics, Speech and Signal Processing ASSP-25*, 6 (December 1977), 554–559.

[3] E.Papamimichalis, P. *Practical Approaches to Speech Coding*. Printice Hall, Inc. Englewood Cliffs, New Jersey, 1987. None.

[4] Glinski, S. C. On The Use of Vector Qantization for Connected-Digit Recognition. *AT&T Technical Journal 64*, 5 (June 1985), 1033–1045.

[5] Itakura, F. Minimum Prediction Residual Applied to Speech Recognition. *IEEE Transactions on Acoustics, Speech, Signal Processing ASSP-23*, 1 (February 1975), 67–72.

[6] Jelinek, F. Continuos Speech Recognition by Statistical Methods. In *Proceedings of The IEEE* (April 1976), vol. 64, pp. 532–536.

[7] Jr., A. H. G., and Markel, J. D. Distance Measures for Speech Processing. *IEE Transactions on Acoustics, Speech, and Signal Processing ASSP-24*, 5 (October 1976), 380–391.

[8] Juang, B. H., and Rabiner, L. R. A Probabilistic Distance Measure for Hidden Markov Models. *AT&T Techinical Journal 64*, 2 (July 1984), 391–408.

[9] Lee, K.-F. *Automatic Speech Recognition*. Kluwer Academic Publishers, 1989. The Development of the SPHINX system.

[10] Levinson, S. E., Rabiner, L. R., and Sondhi, M. M. An Introduction to The Application of The Theory of Probabilistic Functions of Markov Process to Automatic Speech recognition. *Bell System Technical Journal 62*, 4 (April 1983), 1035–1074.

[11] Levinson, S. E., Rabiner, L. R., and Sondhi, M. M. On The Application of Vector Quantization and Hidden Markov Models to Speaker Independent, Isolated Word Recognition. *Bell System Technical Journal 62*, 4 (April 1983), 1075–1105.

[12] Linde, Y., Buzo, A., and Grayx, R. M. An Algorithm for Vector Quantizer Design. *IEEE Transactions on Communications COM-28*, 1 (January 1980), 84–95.

[13] Makhoul, J. Linear Prediction:A Tutorial Review. In *Proceedings of the IEEE* (April 1975), vol. 63, pp. 561–580.

[14] Rabiner, L., and Juang, B.-H. *Fundamentals of Speech Recognition*. Printice Hall,Englewood Cliffs, New Jersey 07632, 1993. None.

[15] Rabiner, L. R. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In *Proceedings of the IEEE* (February 1989), vol. 77, pp. 257–286.

[16] Sambur, M., and Rabiner, L. R. A Speaker Independent Isolated Digit Recognition System. *Bell System Technical Journal 54*, 1 (January 1975), 81–102.

[17] Wilpon, J. G. A Study on The Ability to Automatically Recognize Telephone-Quality Speech from Large Customer Population. *AT&T Technical Journal 64*, 2 (February 1985), 423–449.

[18] Wilpon, J. G., Rabiner, L. R., and Martin, T. An Improved Word-Detection Algorithm for Telephone-Quality Speech Incorporating both Syntactic and Symantic Constraints. *Bell Laboratories Technical Journal 63*, 3 (March 1984), 479–497.

[19] W.Picone, J. Signal Modelling Techniques in Speech Recognition. In *Proceedings of The IEEE* (September 1993), vol. 81, pp. 1215–1247.

Appendix A

Description of Speech Databases

In this appendix we describe the speech databases that we have used in this thesis. We also describe how we collected our own digit database at I.I.T Kanpur, which can be informative for the future research.

A.1 IITK Telephone-Quality Spoken Digit Databases

Recording Setup : We have collected the speech samples from the IIT Kanpur internal telephone network within IIT Kanpur. For recording purpose, we have used a *Zyxel* voice modem on a PC running Linux. The modem supplied *Rockwell ADPCM* compressed data (at $7.2K$ samples per second) and are stored in the raw format. In order to record a speech session, the speaker has to dial the phone number of the modem. As soon as it is connected, the speaker is prompted with a welcome message and is asked to speak after the beep. The speech is then recorded and saved in a file.

Mobilizing The Speakers : We have mobilized around a total of 200 speakers. Each speaker was asked to speak the digits zero to nine in English. Out of these, the recordings for only 163 speakers were good enough and the remaining were discarded.

Recording conditions : Since the speakers are mostly the students calling from hostels and laboratories, the external and background noises are expected. These include noise due to somebody else speaking in the background, noises

due to the running air conditioners and other common noises in the laboratories and hostel corridors.

Speech editing to extract the digits : Initially, we manually edited all the recordings. Later we developed the word detection algorithm and used it. In fact the manual editing experience had been extremely useful in designing the word detection algorithm.

Overall Database Description : This database consists of telephone quality speech of 163 speakers. The first 73 speakers have spoken each digit twice in English and the later 90 speakers have spoken each digit once in English.

Subset Databases : In this section we describe the different databases, which are subset of the above described collection. The names described here are used in discussion throughout the thesis.

IITKdigitSp0to30 : This database is the spoken digit database of the speakers numbered 04 to 30 from a single phone. Each speaker has spoken each digit twice. Each recording contains high amount of noise generated by the equipment. Later, this database was not used and dropped due to high content of noise.

IITKdigitSp31to72 : This database is the spoken digit database of speakers numbered 31 to 72. Each speaker spoke each digit twice.

IITKdigitSp0to72F : This database is a collection of first instance of the digits spoken by the speakers numbered 0 to 72.

IITKdigitSp0to72S : This database is a collection of second instance of each digits spoken by speakers numbered 0 to 72.

IITKdigitSp73to162 : This database is a collection of spoken digits by speakers from speaker 73 to speaker162. Each digit is spoken only once by a speaker.

A.2 OGI Spoken Digit Database

This database is prepared by center for Spoken Language Understanding, Oregon Graduate Institute of Science and Technology.

Recording Setup : This is also telephone quality speech. The speech signal is sampled at 8.0 kHz.

Recording conditions : The recording was conducted under ideal laboratory conditions with no background noise.

Overall Description : This database contains spoken digits of 150 speakers. Each speaker has spoken a digit only once.

Subset Databases :

OGIdigitSp0to119 : This database consists of spoken digits of 120 speakers 0 to 119.

OGIdigitSp120to149 : This database consists of spoken digits of 130 speakers 120 to 149.

OGIdigitSp0to84 : This database consists of spoken digits of 85 speakers 0 to 84.

OGIdigitSp85to149 : This database consists of spoken digits of 65 speakers 120 to 149.