Unifide framework for speaker-aware isolated word recognition

George Rosario Dhinesh

2011

George, R. D. (2011). Unifide framework for speaker-aware isolated word recognition. Master' s thesis, Nanyang Technological University, Singapore.

https://hdl.handle.net/10356/46279

https://doi.org/10.32657/10356/46279

UNIFIED FRAMEWORK FOR SPEAKER-AWARE ISOALTED WORD RECOGNITION

GEORGE ROSARIO DHINESH

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Master of Engineering

2011

Acknowledgement

I would like to take this opportunity to thank all those who have helped me in this research work. First, I would like to express my heartfelt gratitude to my supervisor Dr. Thambipillai Srikanthan for his guidance, patience and understanding. His expert advice during the course of this research work has been invaluable. Despite being entrusted with many responsibilities, he has always found time to discuss the directions for this project with me and I thank him for all the support he gave me.

I would like to thank Mr. Ashish Panda who helped me by explaining the basics of my research work, without which I would have taken much longer time to understand them. His explanation about his previous works was of great help throughout this research work. I thank him for his readiness to help me whenever I had a doubt. I would also like to thank Mr. Jagadeesh for helping me throughout this work. His advice and insights truly helped me at all stages of the research process.

The administrative support rendered by Ms. Nah, Merilyn and Jeremiah was the best I could have asked for. They always helped me out whenever I needed any administrative work to be sorted out. They are truly an integral part of the research lab and their presence is absolutely necessary for the day-to-day functioning. I am also grateful to all of those who made CHiPES a stimulating and interesting atmosphere to work in. Last but not the least, I would like to thank my friends and family for their much appreciated support at all stages of the work.

Contents

List of Figures

List of Tables

Abstract

The explosive growth of various kinds of personal electronic devices in recent years has spawned substantial interest in personalized voice-based human device interaction. There exists a need for robust and computationally-efficient techniques to help realize mobile and embedded computing applications that are capable of recognizing spoken words and the speaker who uttered them. Although spoken word recognition and speaker recognition are closely related problems with a number of commonalities, separate and different techniques are employed for solving them in the current state of the art.

This thesis presents the research, development and prototyping of a speaker-aware isolated word recognition system based on a single, low-complexity technique suitable for resource-constrained mobile and embedded devices. A comprehensive literature survey has been carried out to study and evaluate the suitability of several existing techniques for embedded speaker-and-word recognition. Based on qualitative and performance analyses available in the literature, a framework based on Mel Frequency Cepstral Coefficients (MFCC) and Gaussian Mixture Model (GMM) has been chosen as the base for our work.

An evaluation platform that is rapidly configurable according to the desired values of the parameters involved in the GMM process has been developed in order to expedite the experimentation process. The challenging problem of recognizing a speaker based on a single utterance of very short duration has been examined in detail. The effectiveness of GMM-based text-dependent and text-constrained speaker recognition approaches has been evaluated on the TI46 speech corpus resulting in a recognition accuracy of 99.28% and 96.6% respectively. We have proposed and evaluated a method of grouping similar sub-word units in text-constrained speaker recognition and obtained a recognition rate of 96.62%.

A novel technique has been proposed in order to overcome the inability of GMM to retain the temporal information of the speech in word recognition. This technique relies on modeling a word as a time-ordered sequence of GMMs, where each GMM corresponds to a sub-word unit, so that the sequence of the sub-words is maintained. A simple, yet effective, method with negligible computation overhead has been used to segment words into sub-word units without using any linguistic knowledge. The proposed technique improves the word recognition accuracy by 5.6% when compared to the conventional GMM approach. A further increase of 0.7% in recognition accuracy is observed when multiple models, each with a different number of sub-words, are used to represent a word. Overall, a word recognition accuracy of 96.93% has been obtained on the TI46 database.

Having overcome the limitations that inhibit the use of the conventional GMM approach for speaker and word recognition, the suitability of the proposed technique for embedded applications has been demonstrated by implementing a speaker-aware word recognition system as an Android application on a mobile device. The system has been tested in real-life environments for a closed set of speakers and a limited number of words. The mobile application has been found to correctly recognize both the spoken word and the speaker 91.6% of the time.

Chapter 1

Introduction

1.1 Background and Motivation

With digital devices permeating almost every aspect of modern everyday life, there is an increasing demand for more natural human-machine interaction, improved convenience and better personalization. In line with this trend, the near future is likely to witness a growth of mobile and embedded computing applications that are capable of recognizing spoken words and the speaker who uttered them. For instance, with the advent of ubiquitous computing, it is not far-fetched to envisage a scenario where shared appliances and controls in a home environment respond differently to the same spoken command to suit the predefined preferences of the speaker. Speaker-aware isolated word recognition technology can also find several applications such as personalized educational toys and voice-based multiplayer gaming.

The present approaches towards word and speaker recognition are typically compute-and-memory intensive and do not lend well for implementation in devices having low-speed processors with limited memory. Although there is a substantial body of existing research in the areas of speech recognition and speaker recognition, it is mostly directed towards the problem of improving recognition accuracy without regards to the computational requirement. Such an approach is not sustainable in the age of ubiquitous lightweight computing devices. Hence, there is a need to devise robust and low-complexity solutions for performing speaker-aware isolated word recognition for deployment in resource-constrained mass volume products.

Although speech recognition and speaker recognition typically involve several common computational steps, research on these two closely-related problems have evolved on separate tracks and relatively little work has been done in creating a unified approach. The existing approach to solve the problem of recognizing both the spoken word and the speaker identity is a two-system combo solution, where word recognition and speaker recognition are performed using separate and different methods. If the effectiveness of a single technique is proven to be an efficient alternative to the existing approach, it can facilitate the development of computationally-efficient lean applications well-suited for resource-limited devices.

1.2 Project Objectives

The broad objectives of this research and development work are as follows:

- To perform a comprehensive study of the existing techniques and establish a base framework well-suited for a speaker-aware isolated word recognition system operating on a closed set of speakers and a limited vocabulary. This will involve the optimal selection of features to parameterize speech and an efficient modeling method capable of representing both speakers and words.
- To identify and overcome the limitations of the base framework by proposing innovative improvement techniques and validating them through extensive empirical evaluations on a standard speech database.
- To determine the optimal choice of the various parameters involved in the recognition process through experimentation on the speech corpus.
- To develop an integrated system capable of performing speaker-aware isolated word recognition and to evaluate its effectiveness in real-life applications by implementing it on a resource-limited mobile/embedded device.

1.3 Report Organization

In this chapter, we have established the need for an efficient technique for performing both word and speaker recognition in embedded platforms. The objectives of the project were also defined. In Chapter 2, we present a comprehensive study of the research work in the field of word and speaker recognition. We discuss the complexity and effectiveness of the prevailing techniques for formulating the framework suitable for our purpose.

Chapter 3 describes the base framework consisting of Mel Frequency Cepstral Coefficient (MFCC) feature vectors and Gaussian Mixture Model (GMM). The procedure of GMM-based training and testing, followed in our work is explained in detail. To expedite our research, we have developed a rapidly configurable evaluation platform to carry out all the experiments based on MFCC and GMM. The description and scope of the platform are presented in this chapter.

In Chapter 4, we evaluate the GMM-based text-dependent and text-constrained speaker identification against a standard speech database. We also propose a new method of GMM-based speaker identification based on sub-word grouping. Chapter 5 is devoted to the problem of isolated word recognition. We address a key shortcoming in GMM-based isolated word recognition and propose a sub-word constrained GMM technique to overcome it.

Chapter 6 presents the integrated speaker-and-word identification system. The overall accuracy of the system and the results obtained for different vocabulary sets are presented. This system is ported as an application into a mobile device as described in Chapter 7. Results of the tests, conducted using this mobile application in real-life environments are also presented in the same chapter. Chapter 8 summarizes the major contributions of the project and provides suggestions for future work.

Chapter 2

Literature Review

2.1 Introduction

For an integrated speaker and word recognition system to operate on resource-constrained embedded devices, it has to be low in computational complexity. In this context, a single computationally efficient technique for recognizing both the speaker and the word will be advantageous. A broad review of the literature carried out to identify a potential methodology for the above said purpose is presented in this chapter. The robustness, storage requirement and speed of computations are the main factors that have to be analyzed before selecting a methodology. We start the literature review by discussing the dominant speech feature extraction methods, followed by analyzing the problems of speaker recognition and word recognition separately. Various techniques used for solving both the problems are discussed in addition to the review of the approaches proposed in the literature for realizing an integrated system. At the end of this review, we formulate a base framework and identify the problems that have to be overcome in order to apply it for both speaker and word recognition.

The rest of the chapter is organized as follows. In Section 2.2, we present the speech feature extraction process and discuss two widely used techniques and analyze their advantages and disadvantages. In Section 2.3, we describe the various types of speaker recognition problems and discuss the techniques applied for solving it. Section 2.4 is devoted to word recognition. We study and discuss the various methods used in word recognition and subsequently identify a methodology as the base for our research work. In Section 2.5, we discuss the techniques proposed for reducing the computations in the Gaussian Mixture Model (GMM) method. Techniques proposed in the literature to integrate speaker and word recognition systems are presented in Section 2.6. The contents of this chapter are summarized in Section 2.7.

2.2 Feature Extraction

The first step in both word recognition and speaker recognition systems is the feature extraction process. It obtains the acoustic characteristics of the speech signal. A speech utterance mainly conveys two types of information [1].

- *Linguistic information*: This gives the meaning of the utterance. The main contents of this information are the phonemes, which are a set of sounds in any language.
- *Personal information*: This conveys the identity of the speaker.

The physiological features responsible for speech production suggest that speech individuality is a function of pitch, vocal tract and phonemes [2]. The first two functions refer to the personal information and the last function refers to the linguistic information. A feature extraction process should extract all these information from the utterance. The speech signal is non stationary and it continuously changes due to articulatory movements. Hence, in speech processing the speech signal is broken down into smaller frames of few milliseconds in duration. It is assumed that within this interval, the signal will be stationary and the features are extracted from each frame. In the following subsections, we present two widely used feature extraction techniques and discuss their performance reported in literature.

2.2.1 Mel Frequency Cepstral Coefficients

Mel frequency Cepstral coefficient (MFCC) is one of the most common and widely used methods to represent speech features. The mel-frequency cepstrum is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear scale of frequency known as the mel scale. Mel scale is based on human auditory perception. Psychophysical studies have shown that human ear acts as filter i.e. it concentrates on only certain frequency components and hence human perception of the frequency contents of sounds for speech signals does not follow a linear scale [3]. So the mel scale calculates a subjective pitch for the actual frequency of each tone and it is a linear frequency spacing below 1000 Hz and a logarithmic spacing above 1000 Hz. In MFCC the frequency bands are equally spaced on the mel scale and hence the human auditory system's response is closely approximated.

The calculation of MFCCs from a speech signal is done using a log-energy method and the steps involved in it are given below.

1) The speech signal is converted into frequency domain by Fourier transform.
2) The powers of the spectrum obtained are wrapped into mel scale.
3) Total energy in each mel filter is calculated by adding the square-amplitudes of the points.
4) Logarithm of the energies in each filter is calculated and discrete cosine transformation is applied.

2.2.2 Linear Prediction Cepstral Coefficients

Linear predictive analysis was introduced in the early 1970s and it still remains as one of the most prevailing speech analysis techniques. It is based on the assumption that the variations of vocal tract with time can be approximated, with sufficient accuracy, by a succession of stationary shapes and hence a signal is modeled by a linear combination of its past values and a scaled present input [4] and represented as

$$s(n) = -\sum_{k=1}^{p} a_k * s(n-k) + G * u(n) \qquad (2.1)$$

Here, $s(n)$ is the present output, p is the prediction order, the values of a_k are the model parameters called the linear prediction coefficients (LPC), $s(n-k)$ are the past outputs,

G is the gain scaling factor and $u(n)$ is the present input. The linear predictive approximation, depending only on the past output is

$$\hat{s}(n) = -\sum_{k=1}^{p} a_k * s(n-k) \tag{2.2}$$

The difference between $s(n)$ and $\hat{s}(n)$ is known as prediction error and is given by

$$e(n) = s(n) - \hat{s}(n) = s(n) + \sum_{k=1}^{p} a_k * s(n-k) \tag{2.3}$$

The mean square of the above error E will be a minimum when

$$\frac{\partial E}{\partial a_i} = 0 \qquad \forall i = 1, 2 ... p \tag{2.4}$$

The resulting p equations are solved in order to get the LPC. In [5], it is shown that these equations can be solved by Yule-Walker autocorrelation, covariance or Burg's method. The Linear Prediction Cepstral Coefficients (LPCC) are approximated from LPC by using a conversion formula. The LPCCs smoothen the spectral envelope and give a better representation of vocal tract. The relationship between LPCC c_n and LPC a_k are given by

$$c_1 = a_1 \tag{2.5}$$

$$c_n = \sum_{k=1}^{n-1} (1 - \frac{k}{n}) a_k c_{n-k} + a_n \quad 1 < n \le p \tag{2.6}$$

$$c_n = \sum_{k=1}^{n-1} (1 - \frac{k}{n}) a_k c_{n-k} \quad n > p \tag{2.7}$$

2.2.3 Evaluation of MFCC and LPCC

In the previous two sub-sections, two different methods used to extract features from speech were discussed. In this section, we discuss the advantages and disadvantages of those techniques and their performance. The MFCC features are extracted from fixed interval frame using straightforward signal processing techniques. The log-energy method used here is a direct measure of the energy in different frequency bands. Also, the central frequencies and bandwidths of the mel-filters can be adjusted to match the critical band of the ear. Because of these features the filter energies better capture the perceptually important characteristics of the speech signal [6] [7]. Also it is stated in [7] that while MFCC are not intrinsically robust against health and transmission variances, they lend themselves to some simple compensation techniques to minimize these effects. LPCC is computationally efficient compared to MFCC but LPCs are too dynamic and a small quantization error could cause the entire filters to be unstable and inaccurate. Also, the other drawback of linear predictive analysis is that, it is based on an all pole filter model, which might leave out significant speech spectral characteristics for speech mixed with noise [8].

Davis and Mermelstein [6] tested the effect of five different feature extraction techniques, including MFCC and LPCC, in word recognition. The main conclusion of the experiments was that MFCC gives better performance compared to the other techniques tried. The use of MFCC in many automatic speech recognition systems is established in a survey conducted by Picone [9]. Out of the of 31 reported systems, 21 used some form of cepstral coefficients as the basic signal features, with MFCC the most common type. In speaker recognition problem, Reynolds [10] evaluated and compared several speech representations including MFCC and LPCC and they both showed similar performance. Another comparison of these two techniques in speaker recognition was reported in [11]. The experiments were carried out using clean speech database and telephone speech database. The results show that MFCC performs better than LPCC in the case of clean speech and their performances are comparable for the case of telephone speech.

The above discussion shows that MFCC better captures the characteristics of speech signal and is not in any risk of leaving significant portions of the speech as in LPCC. Also, out of the many comparisons made, it shows similar performance as LPCC and in some cases, gives better performance in both word and speaker recognition. These points indicate that MFCC will be a good choice as speech features for our work. . The detailed description of MFCC calculation is provided in the next chapter.

2.3 Speaker Recognition Systems

Speaker recognition refers to a class of problems where the identity of a speaker has to be decided from his/her voice. This covers both speaker authentication problem and speaker identification problem. In speaker authentication, the speaker claims to be of a certain identity and the voice is used to verify the claim. In speaker identification problem, the speaker does not claim any identity and just offers his/her voice and the unknown speaker has to be identified. It can be said that speaker authentication, also called as speaker verification, is a 1:1 match where one speaker's voice is matched to the claimed speaker's voice model whereas speaker identification is a 1:N match where the voice is compared against N models.

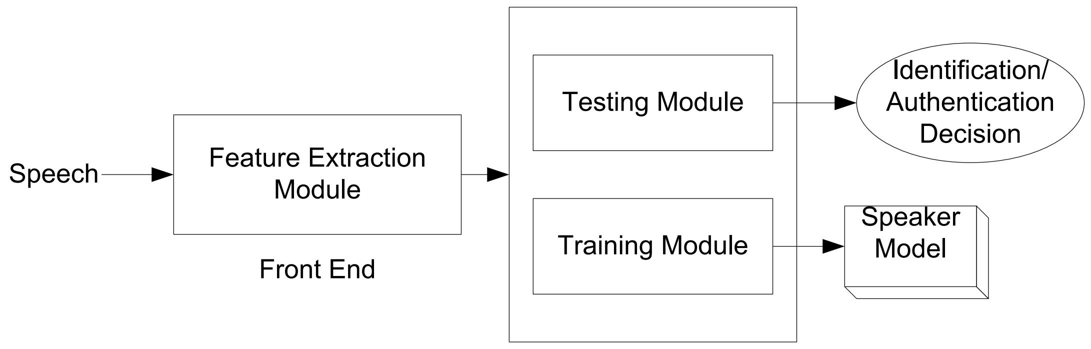
Speech
Feature Extraction Module
Front End
Testing Module
Training Module
Identification/ Authentication Decision
Speaker Model

Figure 2.1: Overview of a speaker recognition system

A speaker recognition system has three basic modules, namely - feature extraction module, training module and testing module. The feature extraction module, as we saw in the last section, extract features from the speech. The training module acts on the speech features and is responsible for preparing a speaker's model that describes the characteristics of that speaker. The testing module can be seen as the decision maker where speaker identification or authentication is carried out using the speech features and the models.

Depending upon the conditions applied to the training and testing module, the speaker recognition problem is further classified into two, text-independent speaker recognition and text-dependent speaker recognition. In text-independent problem, there are no constraints on the words which the speakers are allowed to use during testing. The test utterance can have entirely different content from what was spoken during training. In text-dependent problem, the test phrase is fixed and should be the same used during training.

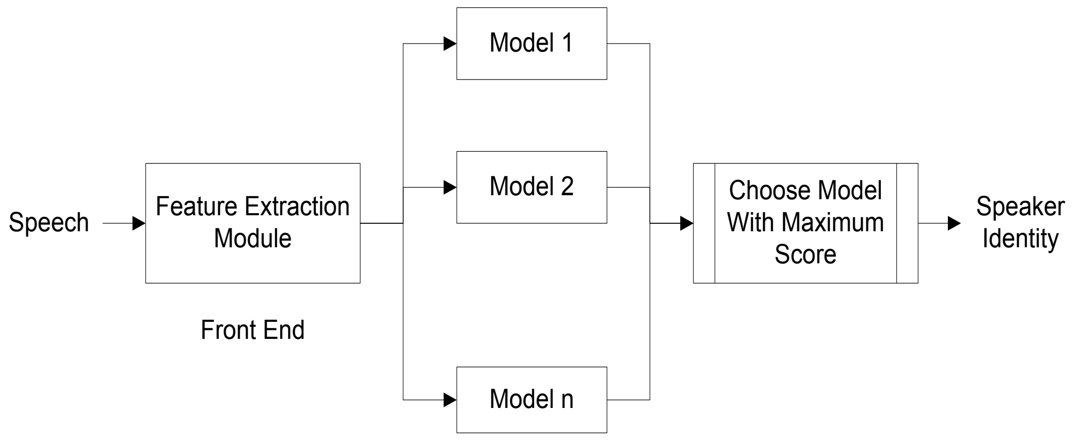

Figure 2.2: Speaker identification process

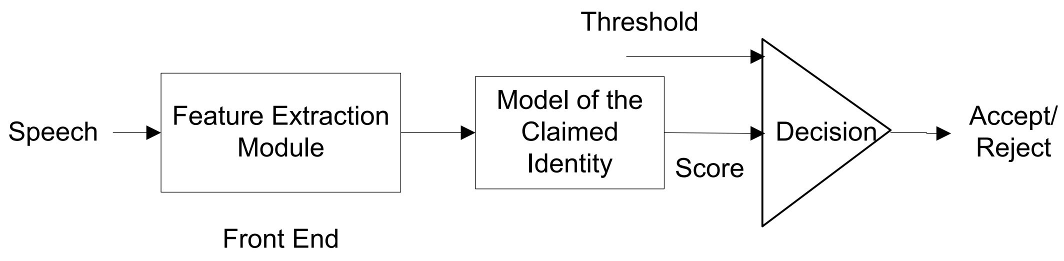

Figure 2.3: Speaker authentication process

The performance measurement of a speaker recognition system depends on the type of problem it deals with. In speaker identification problem, if only a speaker from a set of speakers trained is subjected for testing, the best matching speaker's model is identified as the speaker and the system performance is reported as the number of correct identifications divided by the total number of tests. If speakers outside the training population are also subjected for speaker identification, the identification also depends on a matching threshold. Similar threshold is also used in speaker verification problem to accept or reject the speakers claim and the performance of the system is presented as Equal Error Rate (EER) calculated using the number of false rejections and false acceptance [5, 13].

2.3.1 Speaker Modeling Techniques

In this section, we discuss the widely used techniques in speaker modeling and evaluate their performances. It has to be noted that the modeling scheme determines the testing process as well. The dominant modeling techniques fall into the following two broad categories.

1. Template Based Models: The techniques under these categories are based on the assumption that a test template is an imperfect replica of the reference template of the speaker and the task of recognition is based on the amount of distortion between these two templates. This category includes Vector Quantization (VQ) and Dynamic Time Warping (DTW) techniques.

2. Stochastic Models: Stochastic models express the speaker model in the form of parameters, where the models are described as probability density functions (PDF). If $\vec{x}_i$ is a test vector, the score of $\vec{x}_i$ is given by the likelihood $p(\vec{x}_i \,|\, model)$. This category includes Gaussian Mixture Models (GMM) and Hidden Markov Models (HMM).

2.3.2 VQ Based Systems

VQ was originally used for data compression [14] and was introduced to speaker recognition in the 1980s [15]. A reference template for the speaker, called as codebook, is created using the training feature vectors. For recognizing the speaker, a quantization distortion measure between the test utterance feature vectors and the reference template is calculated using a distance measure such as Euclidean distance. A smaller value of the quantization measure indicates higher likelihood for the test utterance originating from the same speaker. The generation of codebook is the most important part of this technique. Theoretically, it is possible to use all the training vectors directly as the reference template. This increases the size of the codebook and also the calculations in the testing process. Hence, a clustering technique like k-means or Linde, Buzo, Gray (LBG) algorithm is normally used to reduce the number of vectors to the desired codebook size.

In [16], 5 different clustering techniques were experimented in VQ based text-independent speaker identification. It has been shown that choice of the clustering technique is not important. The results obtained for all the clustering techniques were only marginally different. It shows that the easiest way for improving the identification accuracy was to increase the codebook size high enough. The accuracy increased as the codebook size increased from 2 to 64. VQ, though used mostly in text independent speaker recognition, has also been applied for text-dependent speaker recognition as well [17, 18]. The number of training vectors available will be less in this case and hence using a smaller codebook size has been suggested in [17] and using all the vectors as template without clustering is shown in [18].

2.3.3 DTW Based Systems

DTW [19, 20] is an algorithm that uses the principles of dynamic programming to calculate an optimal warping path and overall distortion between two time series. It is the most popular method to compensate for speaking-rate variability in template-based systems [5] and is used in text-dependent speaker recognition. A time-ordered sequence of feature vectors is kept as the reference template i.e. the speaker model. During testing, the test utterance represented as a sequence of feature vectors is scored against the reference templates. In general, even if the same speaker utters the same text, the duration of the test utterance may not be same as the reference template. DTW algorithm normalizes this timing difference by warping the time axis of the test feature vectors to maximize the coincidence with the reference template and then calculates the time-normalized distance, which is the minimized residual distance between the test and reference templates. For a speaker model *i*, this distance is Score(*i*) and a smaller value indicates test utterance originating from the same speaker.

The simplest way to create a reference template is to consider one training utterance as the reference template. Using multiple utterances and applying DTW for training has been suggested in [21]. In that work, one utterance is designated as the initial template, to which a second is time aligned by DTW. The average of the two patterns is then taken to produce a new template, to which a third is time aligned. This process is repeated until all the training utterances have been combined to a single template. For enhancing robustness, multiple templates for speaker models have been suggested in [22]. For testing process, the usual DTW algorithm explained above is not used in this case as it cannot handle multiple templates. A different approach of dynamic programming is applied in that work for speaker recognition and it shows that the multiple templates improve the speaker identification rate by nearly 6% over a single template system.

2.3.4 GMM Based Systems

GMM is a stochastic model and it can be considered as an extension of the VQ model, in which the clusters are overlapping. That is, a feature vector does not belong to one particular cluster depending on the nearest cluster rule as in VQ, but it has a nonzero probability that it belongs to all the clusters. VQ represents a speaker model by a discrete set of means whereas GMM, which is composed of finite mixture of multivariate Gaussian components - also termed as mixtures, represents a speaker by a single mean, weight and a covariance matrix for each component. Determination of these parameters is accomplished by a maximum likelihood formulation which is normally carried out by the iterative expectation-maximization (EM) algorithm [23, 24].

GMM was introduced for speaker recognition by Reynolds [8] in the early 1990s. The use of GMM is motivated by the interpretation that the Gaussian parameters represent some general speaker dependent spectral shapes and the capability of Gaussian mixtures to model arbitrary densities [25]. It is further stated that GMM provides a probabilistic model of the underlying sounds of a person's voice and hence the parameters in GMM reflects some general speaker-dependent vocal tract configurations.

When a GMM is trained, the number of components is predetermined. Mono Gaussian model uses a single Gaussian component and hence have a small number of parameters and therefore computationally efficient, but its accuracy is less than that of multiple-component GMM [26, 27]. The computations in multiple-component GMM can be reduced by keeping the covariance matrix as diagonal. In [28], it has been mentioned that the diagonal matrix outperforms the full covariance matrix. Reynolds and Rose [25] state that the GMM components act together to model the overall probability density function and hence full covariance matrices are not necessary even if the features are not statistically independent.

As a continuation of the GMM approach of speaker recognition, a new system based on GMM and a Universal Background Model (UBM) has been introduced by Reynolds [28]. In this system, a single UBM, which can be considered as a large GMM, is trained to represent the speaker independent distribution of features. This is trained using a collection of speech samples from a large number of speakers representative of the population of speakers expected during recognition. The UBM is trained using the normal EM method, but a speaker model is trained using a form of Bayesian Adaptation (BA) which adapts the parameters of the background model to the feature distribution of the new speaker. During testing, the match score depends on both the speaker model and the UBM. The difference between the score obtained against both the models is considered as the final likelihood score. It has been observed from [29] that this GMM-UBM system provides superior performance over the GMM approach without a background model.

The robustness of the GMM based speaker recognition depends on the amount of speech used for training and the length of the test speech. In [25], various combinations of these two factors have been examined for a text-independent speaker identification system. Training speech amount is varied as 30, 60 and 90 seconds and the test speech length is varied as 1, 5 and 10 seconds. It shows that performance improves when the amount of training and testing speech increases. 10 second test speech duration is shown to give robust results whereas the shorter speech of 1 second length performed poorly even against models trained using 90 seconds of speech. A similar work reported in [30] also shows that a 60 seconds training speech and 10 seconds test speech gives good performance. It is observed that when the amount of testing speech is less the system performance is very low. That is, text-independent speaker recognition is not productive for a shorter test utterance. The GMM approach to overcome this is explained in Section 2.3.6 where we discuss text-constrained speaker recognition.

GMM does not explicitly use any information about sub-words - like phonemes or syllables - from the speech. It pools together the feature vectors of different sub-words while training and hence there is no sub-word alignment between the features of the test utterance and the Gaussian components. Because of the above said non alignment of sub-words in conventional GMM approach, the match score may be biased due to different sub-words in training and test utterances. Few works have been carried out by using GMM for sub-words to overcome this drawback. Text-independent speaker recognition using separate GMM for different syllables has been examined in [31]. In that work, the feature vectors corresponding to a particular syllable in different regions of the speech are used to model that syllable. These set of models are combined at the scoring level using linear logistic regression [32]. In [33], GMM based phoneme modeling and phoneme segmentation of a speaker utterance before recognition is shown to strongly enhance the speaker recognition results. In that work, the phonemes are segmented using a phoneme recognizer and although many phonemes are modeled during training, only few frequently occurring phonemes are considered for testing and still it manages to produce good results. The work reported by Stapert and Mason [34, 35] uses acoustic segments as sub-words and GMMs are trained using a collection of similar segments instead of using the collection of all feature vectors as in normal GMM technique and hence the sequence of feature vectors within the segments are unaltered. The similarity between different segments is measured using the DTW technique. The GMM scoring during testing process is modified to apply to segments rather than feature vectors. This work, carried out to perform text-independent speaker verification with limited training data, preserves the time sequence information of speech and thereby improves the recognition accuracy. The effectiveness of sub-word modeling in GMM-based speaker recognition is worth further investigation.

2.3.5 HMM Based Systems

The Markov process is one where a present event depends on the past events. In a first order Markov process, the state at a particular time *t* depends only on the state at time *t - 1*. The output of the process (observations) is the set of states at each instant of time. In a HMM, transition to a particular state only depends on the state at the previous time instant but the states are not directly observable and the observations are probabilistic functions of the state [36, 37]. An HMM is a doubly embedded stochastic process with an underlying stochastic process that is hidden and can only be observed through another set of processes that produce the sequence of observations [38]. The elements of an HMM are the number of states, the number of distinct observation symbols, the state transition probability distribution which denotes the probabilities of going from one state to another, the probabilities of an observation given a particular state and the initial state distribution which denotes the probability to start in some state.

HMMs are either discrete or continuous. In discrete HMM, the training feature vectors are converted into VQ codebook labels which are in turn used for training the model. During configuration of a continuous HMM, the GMM probability measures obtained from the feature vectors are used. An iterative algorithm known as the Baum-Welch method is used for HMM model estimation [36, 38]. The match score against a HMM model and a feature vector sequence is again calculated as a probability of the vector sequence against the model. There are several variants of HMM depending upon the structure of the state transition probability distribution. Ergodic or fully connected HMM is one where every state can be reached from every other state of the model. To maintain the time sequence of the speech, it is desirable to model the observations in a successive manner. The HMM that fulfills this design is the left-to-right model in which there is no transition from one state to any previous state, thus the model has a time order structure [36, 38].

Ergodic HMM has been applied for text independent speaker recognition [17, 39, 40] and in [40], it has been shown that continuous HMM performs better than discrete one for speaker recognition. It also states that the information on transitions between different states is ineffective for text-independent speaker recognition. Therefore, the speaker identification rates using a continuous ergodic HMM are strongly correlated with the total number of Gaussian mixtures in each state. This indicates that single state continuous HMM, which is nothing but a GMM, can also be used for speaker identification [39, 40, 41]. However, in text-dependent speaker recognition, the information about the time order of the speech is vital and hence left-to-right HMM is preferred [17, 42]. In that case, the length of transition from one state to another is restricted typically to zero (remaining in the same state) and one.

Sub-words based speaker recognition has been carried out using HMM and has yielded good results [43]. The work reported in [44] deals with a similar approach for speaker recognition for digit utterances. It examines two types of sub-words, phoneme units and acoustic units without any linguistic knowledge. It shows that both give comparable performance. Again, a left-to-right HMM is used to capture the time sequence information. The HMM sub-word modeling is also examined in [45] along with whole word modeling for speaker recognition and a hybrid method combining the two is presented to improve the performance.

2.3.6 Text-Constrained Speaker Recognition Systems

There are studies that have looked at improving the performance of text-independent speaker recognition by moving the task closer to the text-dependent case. This was achieved by constraining the speech in the verification process to a limited set of words and is referred as text-constrained speaker recognition in the literature. In the case of text-dependent speaker recognition, the text of the speech can be a single word or a sentence that remains the same during training and testing. In contrast, text-constrained speaker recognition uses a predefined vocabulary of words in training and the test speech can be any word or combination of words from that vocabulary.

Reynolds *et.al* [46] proposed a text-constrained speaker verification system based on GMM-UBM approach. From the training speech, the frequently occurring words are separated using a word-recognizer and grouped into two sets. A separate GMM-UBM model is trained for each of these groups. During verification, only speech from the same group of words as was used to train the GMM-UBM models is used in the likelihood score calculation. It is observed that this text-constrained system gives comparable performance to the baseline GMM approach of text independent speaker verification. A similar work is also reported in [47] where a HMM based approach is followed and instead of word groups, each frequently occurring word is modeled and scoring is done using a forced alignment of the selected words appearing in the test utterance. A HMM based connected word speaker recognition reported in [42] is experimented on a 10 digit vocabulary, where the test utterance is allowed to contain any combination of the 10 digits. Here too, each of the 10 words is modeled separately using HMM and the verification is carried out comparing the test utterance with concatenated HMMs from a designated speaker.

In the above mentioned works on text-constrained speaker recognition, the HMM based approaches use a model for each of the words in the vocabulary. This means the performance of the system also depends upon the accuracy of the extraction of the words from the training speech carried out using the speech recognizer. In the GMM based approach, the words are pooled together and trained. It can be inferred that if only the chosen words are used in training, the extraction process can be eliminated.

2.3.7 Evaluation of Speaker Modeling Techniques

In the previous sub-sections, we discussed about various types of the speaker recognition problem, the widely used modeling techniques and how they are applied to solve it. For a reliable speaker recognition system, the training scheme should result in robust representation of the speaker's voice. From an embedded system perspective, along with this point, the computational complexity and the space required to store the models should also be considered. Also, since the modeling technique determines the testing process, a technique with low-complexity testing procedure will be desirable.

The template based models, especially the VQ method, are computationally simplest and hence suit better than then stochastic models for implementation in embedded devices. However, comparison of performance of VQ based speaker recognition system with stochastic model based systems shows that the later outperforms the VQ method. In [25], GMM based speaker recognition is found to be superior to the VQ based approach. In [40, 17], it is shown that a HMM based system outperforms a system based on VQ. Another template based modeling technique, DTW shows better performance than VQ technique in [17, 18]. DTW has the advantage of non-linearly expanding or contracting the time axis to match between the test speech and reference template [48, 19]. But the implementation increases the cost of storage as the reference template is large compared to other models and it is observed that the idea of keeping multiple templates is suggested for robustness. Also it has to be noted that DTW is widely used only in text-dependent speaker recognition.

Of the two stochastic modeling techniques discussed, GMM has fewer computations as it can be seen as single-state HMM which does not require complex computations to find the state transition probabilities. This means the storage space required by GMM is also less. Reynolds, while introducing GMM for text independent speaker identification states that the models are computationally inexpensive and are easily implementable in real-time platform [8, 49]. Also, successful implementation of GMM based speaker verification system as an application specific integrated circuit is shown in [30, 50]. Unlike GMM, HMM captures the temporal information of speech. However, in the text-independent speaker recognition evaluations carried out by the National Institute of Standards and Technology (NIST), the best GMM based systems have outperformed the HMM based systems [51]. This suggests that in the text-independent case, no gain in performance is being achieved by the use of temporal information captured in the HMM. GMM also shows the potential to carry out speaker recognition using limited test speech by transforming the text-independent problem to a text-constrained one. Sub-word based modeling and testing for speaker recognition has worked successfully in general and using GMM has also shown good results in this area. It is worth investigating if successful GMM based systems that operate on sub-words can be used to further improve the performance of text-constrained speaker recognition systems.

2.4 Word Recognition Systems

Speech recognition is the process of converting a speech signal to a set of words, by means of an algorithm. Speech recognition systems are broadly classified into two different types depending upon the type of speech utterance they can recognize [52]. They are listed in the following.

Isolated Word Recognition: Isolated word recognition systems require the speaker to utter only a single word in the recognition phase. That is, for one test, there is only one word to be identified. Some isolated word recognition systems take a testing sentence with sufficiently long pauses between words and carry out the recognition for all the words in the sentence in isolation.

Continuous Speech Recognition: Continuous speech recognition systems allow users to speak sentences in the natural fashion without any constraints and determine the content of the speech.

The continuous speech recognition is a complex problem where it has to deal with determining the word boundaries and identifying the words from the large vocabulary in the language spoken. It is clear that the isolated word recognition, which is the focus of our project, is the simplest speech recognition problem where the only task is to identify a word uttered.

2.4.1 Overview of Isolated Word Recognition System

The overall structure of an isolated word recognition system matches that of a speaker recognition system shown in Figure 2.1. The feature extraction process is the front end for both training and testing. A model for a word is obtained as the output of the training process and the testing process takes one word as input and finds its match using the registered word models. The main factors that affect the performance of an isolated word recognition system are the vocabulary size and the speaker dependency.

Vocabulary Size: Isolated word recognition systems are generally classified as small or large vocabulary depending on the number of words it can identify. Though there is no standard size for these classifications, it has to be noted that when the vocabulary size is large, the memory required for storing all the word models will become high and input utterance should be matched with all the word models to choose the best, which will increase the computations. These drawbacks have been overcome by using models for sub-word units, like phonemes or syllables, instead of whole word modeling. The test speech sub-words are compared with those models and the results combined with a word dictionary gives out the output word. In this case, when the vocabulary needs to be expanded, words can be added to the word dictionary and the phoneme template need not be changed. Hence the memory required and the computations do not increase as much as the whole word modeling case.

Speaker Dependency: A word recognition problem can also be classified as speaker dependent or speaker independent. This classification depends upon the mode of training. A speaker-dependent word recognition system uses only the utterances of a single speaker to model the words. The system is then specifically used for recognizing the word uttered by the same speaker. This speaker-dependent system can be extended from a single speaker to a group of speakers as well. A speaker-independent recognizer is used to recognize the words from speakers who may even be outside the set of speakers used for training. In this type, the word models are generally prepared using utterances from large number of speakers. The speakers should span a wide range including ranges in age group, accent, gender and speaking rate [53].

Like the speaker recognition modeling techniques described in section 2.3.1, the techniques used in word recognition can also be classified as template based models and stochastic models. The VQ and DTW that fall under template based models category are the earliest techniques used in solving the word recognition problem. The HMM based stochastic models are also widely used in solving the problem. In the following, we discuss how these techniques are applied for word recognition based on full word modeling and sub-word modeling.

2.4.2 Word Recognition Based on Whole Word Modeling

Initial approaches in word recognition followed the method of modeling the entire word and thereby having a model or reference template for each word in the vocabulary. VQ is one of the earliest techniques applied to isolated word recognition. In [54], it has been applied in word recognition with a vocabulary of 20 words and a recognition accuracy of 99% is obtained for speaker-dependent case and 87% for speaker-independent case. The reason for the poor performance in the second case is attributed to the omission of time alignment in VQ modeling. DTW overcomes this problem and as mentioned earlier, it compensates the speaking rate variability, which is an important factor in speaker independent word recognition, by nonlinearly expanding or contracting the time axis to match the test utterance and reference templates.

The DTW based word recognition systems rely heavily on the robustness of the reference templates. For preparation of better reference template for a word, a technique called crosswords reference templates has been proposed in [55]. In that work, one utterance is selected as initial template. Then multiple utterances of the word are considered and these utterances are time aligned using DTW with the initial template. The final template is taken as the average of all the time aligned utterances. This approach is shown to give almost 14% increase in word recognition accuracy compared to the DTW technique where just one utterance of a word is directly used as the reference template. Another method to improve the robustness is to use multiple templates for each word. In this case, the matching of a test utterance against a word is carried out by applying DTW to all the reference templates of the word and selecting the best one or averaging the match scores.

Currently, the most successful and widely used approach for speech recognition is HMM. In speech recognition, it is desirable to use a model which models the speech in a successive manner because it resembles the property of speech. Of the various types of HMM, the left to right HMM is the one that fulfills this technique and hence it is used in the speech recognition field [52]. The whole word modeling based on HMM represents each word with a certain number of states. Like using multiple reference templates for DTW based word recognition discussed above, multiple HMM for each word has been proposed in [56]. This approach has been applied to speaker-independent word recognition and a reduction of about 50% in error rate is obtained in comparison with a single model system.

2.4.3 Word Recognition Based on Sub-Word Modeling

A word recognition system that is based on sub-word modeling has two major components, the reference templates or models for the sub-word units and the word dictionary. Phonemes are mostly used as the sub-word units and the words in the dictionary are represented as a sequence of phonemes. The number of phonemes used in the systems of various languages is around 40 to 50. The incoming test utterance is matched against all these phoneme models to find the sequence of phonemes. This process forms the first step in word recognition and the output is matched against all the words in the dictionary to identify the word. The usage of HMM for this type of word recognition is widely found in the literature [57, 58, 59, 60].

In a phoneme-based word recognition system using DTW, a different approach to the above mentioned one is used [61]. In this system, phonemes are not determined at the first step, but a similarity or distance values between each frame of phone reference template are used to identify the word from the dictionary. The similarity values are stored in a matrix and the accumulated similarity between each word in the vocabulary and the test utterance is calculated using the values in the matrix.

2.4.3 Evaluation of Word Recognition Systems

In word recognition systems, the choice of whole word modeling or phoneme based modeling depends on the size of the vocabulary. In phoneme based word recognition systems, it has to be noted that since each language has around 50 phonemes, there will be as many models in the system and also, there is a second step where the matching of the phoneme sequence with the words in the dictionary has to be carried out. So it is clear that for a word recognition system that has to deal with a vocabulary of size around 50, whole word based modeling can achieve the output with similar computations and memory space as that of phoneme based modeling. In whole word modeling, it has been observed that using multiple templates or models for each word has improved the performance of the system using one template or model per word. Though it increases the computations and the storage space marginally, it should not be a major setback as all the systems using whole word modeling deal only with a small set of words.

In the previous section, we discussed about speaker recognition systems and found that GMM is more robust and it has low computational complexity among the stochastic models. However, in word recognition, it is observed that GMM based modeling is not widely used. This is mainly because of the lack of temporal information in GMM models. Incorporating time sequence information in GMM models can potentially make it suitable for word recognition. This concept of adding time information has been applied to VQ technique in the form of spectral-temporal codebooks for improving the word recognition [62]. Also, as discussed in the previous section, for speaker identification, sub-word based recognition has been carried out successfully using GMM. Therefore, if GMM is made suitable for word recognition through the incorporation of temporal information, it can lead to a successful low-complexity integrated speaker and word recognition system based on a single methodology.

2.5 Alternative Modeling and Testing Schemes for GMM

In the previous two sections, we discussed various methodologies for speaker recognition and word recognition and identified GMM as the potential methodology for our research work. Though we observed earlier that the computational complexity involved in GMM is not high, alternative approaches for further reducing the computations and thereby speeding up the GMM based systems will be of beneficial. In this section, we present such works reported in the literature.

2.5.1 Alternative Modeling Schemes

In GMM, the conventional method for training a model uses the Expectation Maximization (EM) algorithm [63]. The EM algorithm involves many computationally intensive operations like square root, exponentiation and division. Also, to guarantee convergence to a local maximum, it usually takes 10 iterations. This means that the number of intensive computations involved grows exponentially with the number of training vectors and linearly with the number of iterations. Also, the EM algorithm acts on an initial model, which has to be calculated using a different algorithm. This adds to the cost of computations and hence, it consumes more time in the training [49, 64]. It is clear that because of these drawbacks, the implementation of EM algorithm would be expensive and difficult in resource-constrained devices. Hence, there is a strong need to look into alternative schemes that can reduce the computational complexity present in GMM training. In this section, we discuss such schemes reported in the literature.

The vector quantization clustering techniques like k-means algorithm and Linde, Buzo, Gray (LBG) algorithm have been tried to replace the EM method in [65]. The k-means algorithm [66] is a clustering algorithm which represents each cluster by the mean of the cluster. Assuming a set of vectors $X = \{\vec{x}_1, \vec{x}_2, ..., \vec{x}_T\}$ is to be divided into M clusters represented by their mean vectors $\{\vec{\mu}_1, \vec{\mu}_2, ..., \vec{\mu}_M\}$, the objective of the k-means algorithm is to minimize the *total distortion* given by

$$total\, distortion = \sum_{i=1}^{M} \sum_{t=1}^{T} \| \vec{x}_t - \vec{\mu}_i \| \tag{2.8}$$

The k-means algorithm follows an iterative approach to meet the objective. In every iteration, it redistributes the vectors in order to minimize the distortion. LBG algorithm [67] shares many of the characteristics of the k-means algorithm. Like k-means, it too was developed originally for vector quantization purpose and aims to minimize the total distortion. However, unlike k-Means, LBG does not estimate the means of all M clusters in each iteration. Rather, it starts with a single cluster and arrives at M clusters by splitting it. With each successive iteration, the number of clusters is doubled. Thus the final number of clusters M could only be a power of 2.

The performance of k-means and LBG algorithm for training GMM is examined and reported in [65]. The two algorithms were used in a text-independent speaker recognition experiment by training a GMM with 32 components from 6000 vectors of 20 dimensions each. The experiments were carried out using two speech databases, KING [68] and TIMIT [69] and it has been reported that the complexity of EM algorithm is nearly twice the complexity of k-means and LBG algorithms. Hence the computational complexity involved in training GMM is reduced by half using k-means and LBG algorithm. But performance wise it has been shown that accuracy of EM algorithm is slightly more than both k-means and LBG algorithms. For the KING database, EER of 1.4% is obtained for EM method where as for k-means and LBG method it is 2.0%. For TIMIT database, the EER obtained is 0.39%, 0.47% and 0.5% for EM, k-means and LBG algorithms respectively.

An alternative training scheme based on Bayes Adaptation (BA) technique is proposed for GMM based speaker verification system in [30]. BA is a non-iterative method and has two major steps in it namely, the expectation step and the adaptations step. The expectation step is similar to the EM algorithm steps and the adaptation step determines the extent to which a particular component or mixture can be altered. BA has been used in [28] to adapt speaker models from a universal speaker independent model. But in [30], it has been proposed to employ BA in a different manner. BA is used on the initial model estimated by K-means algorithm to get the final model. The initial model estimated by K-means could be thought of as speaker-dependent prior parameters. So, in other words, the BA has been used to estimate the final model from the speaker-dependent prior parameters rather than the speaker-independent parameters. The comparison of this technique with EM, for text-independent speaker verification on KING and TIMIT database, carried out in [30] shows that the time taken for training a speaker model, on 1 minute speech, using KING Database for EM algorithm is 111 seconds, while the time taken for BA training is 12.3 seconds. On TIMIT database, for 24 seconds speech, the training time for EM algorithm and BA was 44.1 and 4.5 seconds respectively. These results indicate that BA scheme is nearly 10 times faster than the EM algorithm scheme. The computational complexity of BA is shown to be nearly equal to the computational complexity of one iteration of the EM algorithm. Also, it has been reported that For KING database, EER of 1.4% is obtained for EM method where as for BA method it is 1.35%. For TIMIT database, the EER obtained is 0.39% and 0.41% for EM and BA methods respectively.

From the above discussions, we can observe that the performance of BA technique better the two vector quantization techniques and is comparable to that of EM algorithm. The main advantage identified is that BA technique reduces the computational complexity by a factor of 10 and if a system prefers low computations at the cost of negligible reduction in accuracy, then it would be an ideal choice. In our work, this method will be of great benefit in keeping the computational complexity low as both word and speaker models should be trained.

2.5.2 Alternative Testing Schemes

In GMM based systems, the computations in the testing process depends on the number of mixtures present in the model and the length of the test speech i.e. the number of feature vectors used for testing. Works have been reported about reducing the computations to speed up the output of the testing process. The work carried out in [70] proposed a method based on Gaussian component selection to reduce the computational effort involved in the speaker verification process. The method speeds up the scoring process by considering only those GMM components that are likely to obtain good scores. A smaller hash model is created to index the likely components in the base model. It shows that scoring only 128 out of 1024 mixture components of a GMM model leads to no noticeable performance degradation. This reduces the processing required by a factor of 8. A further reduction down to 32 components only leads to minor degradation of performance. McLaughlin *et al*. [71] have studied two computational speed-up methods for the GMM/UBM based speaker verification system, decreasing the UBM size and decimating the sequence of test vectors. It has been noticed that the large UBM could be reduced by a factor of 4 without degrading the performance. Another study in that work shows that the reduction of the feature vectors by a factor of 20 during testing does lead to any loss in the speaker verification accuracy.

Apart from the length of feature vectors and the number of mixtures in the model, the number of models against which the scoring has to be done for identification also increases the computational time required for the result. A few works have dealt with speeding up GMM based speaker identification carried out against a large set of speakers.

In [72, 73], a hierarchal structure of all the speaker's models is proposed by repeating the merger of the closest GMMs until the number of GMM becomes 1. This method has been shown to speed up the speaker identification for a set of 40 speakers by a factor of 3 with almost similar identification accuracy. Pellom and Hansen [74] presented a computationally-efficient GMM based speaker identification system by applying beam-search pruning and reordering the feature vector sequence. The sequence vectors are reordered so that the non adjacent feature vectors are scored first. This is based on the concept that for speaker recognition, time order of the feature vectors is not important and the adjacent vectors are correlated. After the first scoring, a beam search technique is used to prune out the worst scoring speaker models. This is followed by further reordering of the feature vectors. These two processes are repeated as long as there are speaker models or feature vectors left, and then the best scoring model is selected as the identified speaker. This work reports a speed-up factor of 6:1 relative to speaker identification from a set of 138 speakers using the baseline beam search.

The above discussion covered a few points on speeding up the testing process. It is observed that if the number of mixtures in a model is large, it is not necessary to score against all the mixtures but only against a selected set of mixtures likely to yield good scores. However, when a small number of mixtures are used, leaving out any mixture could result in loss of vital data and reduce the performance. Similarly reducing the size of the feature vectors has been carried out only against a large UBM and not against any model with few components. For identification of speaker, it has been observed that when the number of registered speakers is high, the normal approach of scoring against all the models is time consuming and following alternative methods is productive.

2.6 Integrated Speaker and Word Recognition Systems

In this section, we discuss the integrated speaker and word recognition systems reported in the literature. Though many works involving both speaker recognition and word recognition has been reported, only a few works have been aimed particularly at identifying both the speaker and the word. One such work has been carried out by Reynolds and Heck [75]. They proposed a system by combining a text-independent speaker recognition based on GMM and a speech recognizer based on DTW. The description of the system is presented below in Figure 2.4.

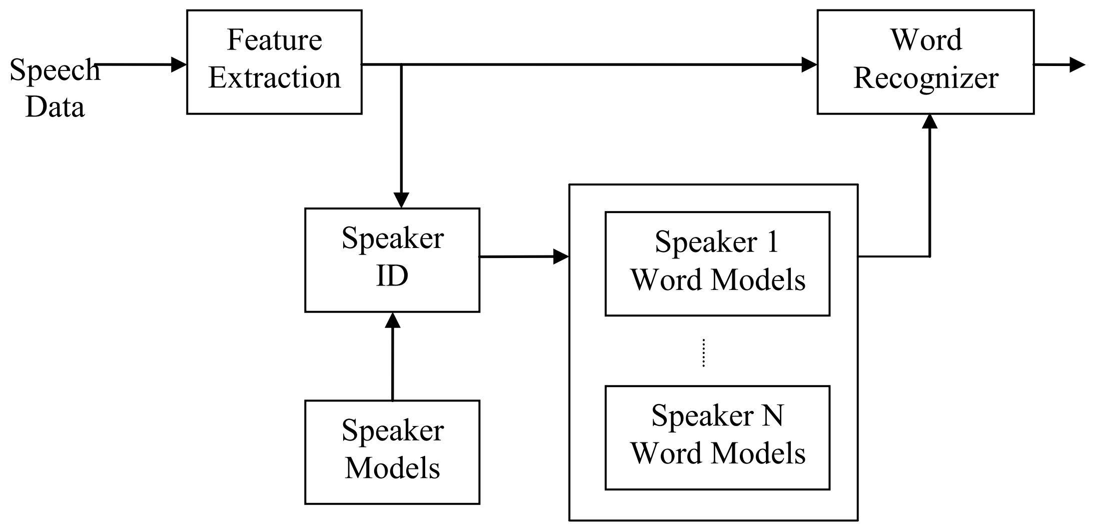

Figure 2.4: Integrated speaker and isolated word recognition system proposed in [75]

The speaker recognition system is the first process where the speaker is identified using the utterance of an input word. The speaker identity is then used to choose the reference word models for the speech recognizer. The experiment is carried out for a vocabulary of 10 words which means for each speaker, there will be 10 speaker dependent word models. The word recognition setup is based on the established rule that speaker-dependent recognition yields superior performance. It has been suggested that for speakers outside the registered set of speakers, the speaker identification part can be considered as a 'speaker quantizer' which associates the unknown speaker with an acoustically similar speaker for better word recognition.

Like the concept of speaker quantizer mention above, few integrated systems use speech quantizer to improve the performance of speaker recognition. In [76], robust text-independent speaker recognition is achieved by using a DTW based word recognizer as front end. DTW word spotting finds the words in the test speech that resemble the words in the training set and only those words are used for testing. In [77], automatic speech recognition provides a phonetic segmentation of the test utterance and thereby facilitating strong phoneme based speaker recognition. The text-constrained systems we discussed in Section 2.3.6 [46, 47] also combines a speech recognizer to find the frequently occurring words to use it for text constrained speaker recognition.

Study of various integrated systems shows that none uses an efficient single methodology that independently recognizes the speaker and the word. However, this is by no means infeasible because techniques such as DTW and HMM have been reported to be successful in solving both word and speaker recognition. However, we have identified that from an embedded system perspective, GMM would be an ideal choice. New techniques have to be devised to make GMM suitable for isolated word recognition. It is observed that in integrated systems, using one recognizer as front end can improve the performance of the other. A word detector front end can make the second step as text-dependent speaker recognition and a speaker recognizer front end can facilitate speaker-dependent word recognition. This point could also be utilized to improve the performance of the integrated system based on GMM.

2.7 Summary

In this section, we summarize the major findings from the literature survey and on their basis, outline some thoughts related to our work. In the first part of this chapter, we examined the feature extraction techniques. We analyzed two widely used techniques for extracting speech features, MFCC and LPCC. MFCC, based on human auditory perception, has been shown to capture the speech characteristics better and it has been found that in both word and speaker recognition, it outperforms LPCC in many experiments. Hence, we have decided to use MFCC as speech features for our work.

We presented a comprehensive report of various methodologies used in speaker recognition. It is observed that GMM is the state-of-the-art method for speaker recognition in the text-independent scenario. Various researches in that scenario show that 5 to 10 seconds of speech is needed for successful testing of the speaker, whereas in our work, only one utterance of a word, typically of one second duration or less, will be available. This difficulty can potentially be overcome by moving the problem from text-independent domain to text-constrained domain. We discussed how a GMM based speaker recognition system that confines the training and testing processes to some frequently occurring words from a continuous speech input, has resulted in good recognition performance. For our work, it will be worth investigating the performance of the text-constrained approach, for even shorter test utterances.

Compared to other modeling techniques, GMM is less complex in terms of both computation and storage size. Techniques that further reduce the computational complexity in GMM have also been reported. We discussed how BA based GMM modeling can simplify the computations in the conventional approach. In spite of these advantages, the review of isolated word recognition systems shows that GMM has not been widely used in word modeling. It is found that word recognition systems depend on modeling techniques that capture the temporal information of speech and GMM falls short in that aspect.

Sub-word based word modeling methods have been reported in literature and they are preferred over whole word modeling techniques, for large vocabulary word recognition. It is observed that sub-word modeling and testing based on GMM has been successfully used in the field of speaker recognition. Hence, for word recognition, the effect of using GMM in the context of sub-words is worth investigating and a successful outcome can lead to a single modeling scheme for an integrated system of speaker and word recognition. We also discussed various systems proposed to integrate speaker and speech recognition tasks. Most of the systems use two different methodologies for the two tasks which increases the computational complexity. From these systems, it is observed that one task can be considered as a front end and its output can be used to improve the performance of the other task.

In the following chapters, we propose new techniques for GMM based speaker and word recognition and present the results. We foresee that exhaustive evaluations will be required in our work and hence a rapidly configurable evaluation platform is developed to speed up the research. In the next chapter, we provide the description of this evaluation platform and explain its capacity to assist any research work related to this field.

Chapter 3

Platform for Evaluating GMM Based Experiments

3.1 Introduction

In the last chapter, after a comprehensive study of various techniques involved in speaker and word recognition, we concluded that GMM with MFCC as features of the speech will be the ideal scheme that best suits our objective of developing a low computational complexity algorithm for speaker and word recognition. A GMM-based recognition system, as described in detail later in this chapter, has several process parameters such as the number of mixtures and the dimension of the feature vectors whose values determine the robustness of the model. The best system performance can be established only after analyzing various combinations of values of these parameters. Our research requires these exhaustive evaluations in addition to trying modifications of conventional GMM approach for modeling a word or speaker model.

Although software toolkits exist for the standard tasks involved in speech signal processing and statistical modeling, they typically require user intervention between the steps and do not lend well for fully automated execution of the experiments, which is necessary while working with a large set of data. Besides, our research requires the ability to completely customize and modify the processes involved. While off-the-shelf toolkits are adequate for implementing standard computational blocks, it is often hard to carry out extensive modifications to them. As such, it is advantageous to custom-build an evaluation platform tailored towards meeting the needs of this project. Hence in order to expedite our research work, we intend to build such an evaluation platform that allows the user to effortlessly and rapidly select a methodology, specify the values of various parameters and perform an evaluation quickly on one among many speech databases. It is envisaged that this evaluation platform will serve as a useful tool not only for our project but also for future research on speaker and word recognition. In this chapter, we present the detailed description of the architecture of the evaluation platform and discuss how it can be easily configured to run an experiment.

The rest of the chapter is organized as follows. In Section 3.2, we describe the steps involved in extracting the MFCC as feature vectors. In Section 3.3, we discuss the GMM concept. As mentioned in the last chapter, we have chosen the k-means algorithm for model initialization and a variant of the Bayes Adaptation [30] method for modeling. These methods are explained in this section. In Section 3.4, we explain the evaluation platform in detail and discuss the procedure for using it to run an experiment. Section 3.5 concludes this chapter.

3.2 Mel Frequency Cepstral Coefficients (MFCC)

Human auditory system does not perceive frequency of a tone in a linear manner [52]. The peripheral auditory system behaves as if it contains a bank of band pass filters with overlapping pass-bands. The widths of these conceptual filters are known as critical bands. It is due to these filters, human response to frequency is non-linear. Mel is a unit of measure of this perceived frequency or pitch of a tone. Mel scale is approximately linear below 1000Hz. However, it becomes logarithmic above 1000Hz. The mel value could be approximated from the frequency in Hz by the following equation

$$F_{mel} = \frac{\ln\left(1+\frac{F_{Hz}}{700}\right)*1000}{\ln\left(1+\frac{1000}{700}\right)} \tag{3.1}$$

In Equation 3.1, F_{mel} is the frequency in mel scale and F_{Hz} is the frequency in Hz scale. The mel effect is simulated by overlapping triangular band pass filters. These filters are placed linearly in the range of 0 to 1000Hz and logarithmically afterwards. To calculate

MFCC using these mel filters, a log total-energy method is used where all the points in the speech spectrum are considered and the total energy obtained in each of the filter is used. The steps involved in MFCC calculation are summarized below and illustrated in Figure 3.1.

Hamming Window: In signal processing, a window function is one that is zero-valued outside of some chosen interval. When another function or a signal (data) is multiplied by a window function, the product is also zero-valued outside the interval. Hamming window is one such window function and is suggested as the best one for speaker recognition in [5]. Each window corresponds to one speech frame. The speech values are multiplied with the Hamming window to smoothen the speech signal by minimizing the signal discontinuity. The Hamming window in discrete time sequence is used as follows [78]

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

Fourier Transformation: For faster computation, Fast Fourier Transformation (FFT) is used. If the number of points in the speech frame is not a power of 2, then zero padding is used.

Mel-filter Bank Application: The triangular mel filters are applied to the speech frame and the total energy in each filter is calculated by adding the square-amplitudes of the points.

Log-Discrete Cosine Transform (DCT): After calculating the total energy in each of the mel filters, log and inverse Fourier transformation are applied to the total energy in each filter to obtain the mel frequency cepstral coefficients. Since energy is an even function, the inverse Fourier transformation has only the cosine terms. Thus Discrete Cosine Transformation (DCT) can be used in the place of inverse Fourier transformation.

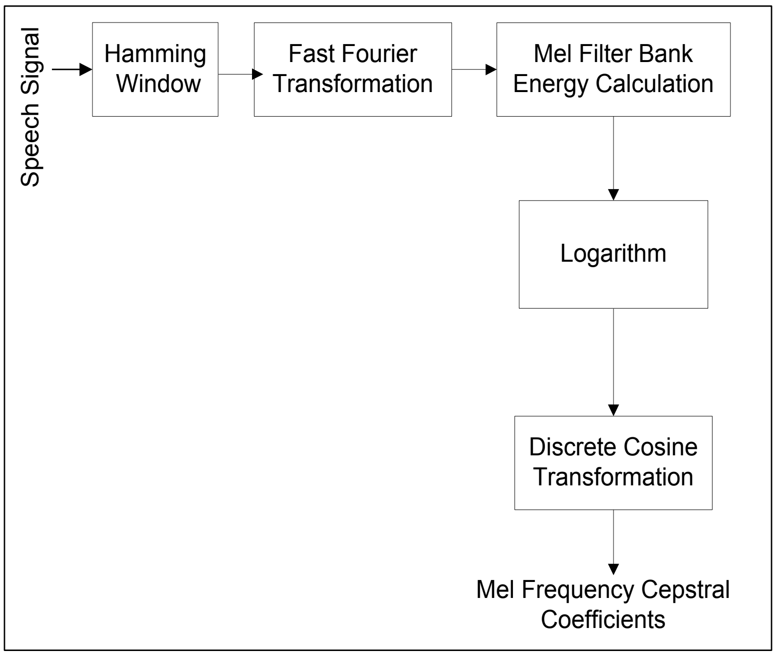
Speech Signal
Hamming Window
Fast Fourier Transformation
Mel Filter Bank Energy Calculation
Logarithm
Discrete Cosine Transformation
Mel Frequency Cepstral Coefficients

Figure 3.1: Mel Frequency Cepstral Coefficients Calculation

In this section, we have described the concept of MFCC and the steps involved in extracting it from speech data. The distribution of these MFCC vectors in the form of a GMM is used to model a word or speaker. The principle of GMM and the salient issues involved in it are discussed in the next section. Also, the technique followed in our work to train the GMM is explained in detail.

3.3 Gaussian Mixture Modeling (GMM)

Mixture distribution is a term in statistics, used to express a distribution that can be represented as a superposition of component distributions [63] and the component can be any probability density function (PDF). The Gaussian distributions are widely used as components due to the fact that, virtually any real world distribution can be represented as a mixture of Gaussian distributions [79]. The PDF of one-dimensional Gaussian distribution, also known as one-dimensional Normal distribution, is given by

$$b(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{3.3}$$

where, μ is the mean and σ^2 is the variance of the distribution. If we extend the above equation to *D*-dimension, then the PDF of *D*-dimensional Gaussian distribution [80] will be given by

$$b(\vec{x}) = \frac{1}{(2\pi)^{D/2} |\Sigma|^{1/2}} \exp\{-\frac{1}{2}(\vec{x}-\vec{\mu})^T \Sigma^{-1}(\vec{x}-\vec{\mu})\} \quad (3.4)$$

where, $\vec{\mu}$ is the mean vector (D-dimensional quantity) and Σ is the covariance (*D* x *D*) matrix. Mixture of Gaussians is a linear combination of Gaussians (Normal Distributions), either one-dimensional or multi-dimensional. So a mixture of Gaussians [63] with *M* mixture components (i.e. *M* Normal Distributions), with *D* dimensions each, can be represented as

$$\sum_{i=1}^{M} w_i b_i(\vec{x}) \quad where,\ b_i(\vec{x}) = \frac{1}{(2\pi)^{D/2} |\Sigma_i|^{1/2}} \exp\{-\frac{1}{2}(\vec{x}-\vec{\mu}_i)^T \Sigma_i^{-1}(\vec{x}-\vec{\mu}_i)\} \quad (3.5)$$

The term w_i in the above equation is known as the weight and $\sum_{i=1}^{M} w_i = 1$. As with the normal distribution, mixture of Gaussians can also be completely represented by the weights, mean vectors and covariance matrices of the mixture components. A model, which models a process as a mixture of Gaussians, is called a Gaussian Mixture Model (GMM). We denote a GMM as λ and it can be represented as

$$\lambda = \{w_i, \vec{\mu}_i, \Sigma_i\} \quad i = 1,2,..,M \quad (3.6)$$

In the following subsections, we describe the methods followed in our work to solve two important problems associated with GMM before it can be used to represent a speaker or word. The two problems are listed below.

1) Given the observation vectors $\{\vec{x}_1, \vec{x}_2, ..., \vec{x}_T\}$, estimation of the model $\lambda = \{w_i, \vec{\mu}_i, \Sigma_i\}, \quad 1 \le i \le M$, which best explains the observation vectors. This

can be seen as the training problem. That is, given the training sequences, create a model for the word or speaker.

2) Given the test vectors $\{\vec{x}_1, \vec{x}_2, ..., \vec{x}_n\}$ and the model $\lambda = \{w_i, \vec{\mu}_i, \Sigma_i\}, \quad 1 \le i \le M$, calculation of the score of the test vectors with the model. This problem can be seen as the recognition problem i.e. with some trained models, each model representing a word or speaker, which model is the most likely one if an observation is given?

3.3.1 Estimation of Model Parameters

For estimating the model parameters of a GMM, out of the many methods available, maximum likelihood estimation is the most practical and widely used method [6]. Suppose, $\{\vec{x}_1, \vec{x}_2, ..., \vec{x}_T\}$ is the set of observation vectors and λ is the model. Then the likelihood L of the observation set given the models is

$$L = \prod_{i=1}^{T} p(\vec{x}_i \mid \lambda) \tag{3.7}$$

The goal of maximum likelihood estimation is to maximize L by adjusting the parameters of λ. This is accomplished by Expectation Maximization (EM) Algorithm. EM algorithm is an iterative procedure, which, given an initial model, provides better approximations with each iteration. However, in our project we have decided to use a method proposed in [30] which is based on the Bayes Adaptation (BA) technique [28]. As mentioned in the literature survey, the high speed and low computational complexity of this technique are the reasons for us to use this method. The first step in BA is identical to the EM algorithm. In addition, in the second step, BA combines the statistics from a universal background model with the new statistics obtained from the first step to estimate the final parameters. In our training method, BA is used on the initial model instead of a universal background model to obtain the final model. Before we explain the BA based training scheme in detail, we describe the k-means algorithm used to estimate the initial model.

Initial Model Estimation: In [25], three different model initialization techniques, including the k-means algorithm, have been evaluated to find its effect on the performance of GMM and it has been stated that the initialization techniques have no effect on the performance of GMM. Hence, without further examination of the techniques, we have decided to use the k-means algorithm to initialize the model in our work. K-means algorithm [66], which follows an iterative approach, was originally designed for vector quantization codebook generation. It is a clustering algorithm which represents each cluster by the mean of the cluster. If a set of vectors $X = \{\vec{x}_1, \vec{x}_2, ..., \vec{x}_T\}$ is to be divided into M clusters represented by their mean vectors $\{\vec{\mu}_1, \vec{\mu}_2, ..., \vec{\mu}_M\}$, the objective of the k-means algorithm is to minimize the *total distortion* given by

$$total\,distortion = \sum_{i=1}^{M} \sum_{t=1}^{T} \left\| \vec{x}_t - \vec{\mu}_i \right\| \tag{3.8}$$

This algorithm is adapted to initialize the GMM with *M* mixture components and the steps involved in it are described below and illustrated in Figure 3.2.

1) To initialize, M random vector from the training set are selected as the means of M mixtures.
2) Each vector $\vec{x}_t$, $1 \le t \le T$, is assigned to mixture j, *iff*, $\left\| \vec{x}_t - \vec{\mu}_j \right\| < \left\| \vec{x}_t - \vec{\mu}_k \right\|$, $\forall k \ne j$, $1 \le j, k \le M$.
3) The new mean of a mixture is obtained by calculating the mean of all the vectors assigned to that particular mixture.
4) The weights are determined by calculating the proportion of the vectors assigned to the mixture and the covariance matrix is the covariance matrix of the assigned vectors.

Steps 2 and 3 are repeated till the mixtures are stable, i.e., the distortion is minimized and is less than a threshold. When the mixtures are stable, the weight and covariance matrix can be found out as described in Step 4. It is to be noted that in each of the iterations, k-means estimates the means of all the M mixtures. In [50] 10 iterations of k-means algorithm were recommended for the distortion to get minimized. In our work also we follow 10 iterations of k-means algorithm to initialize the model.

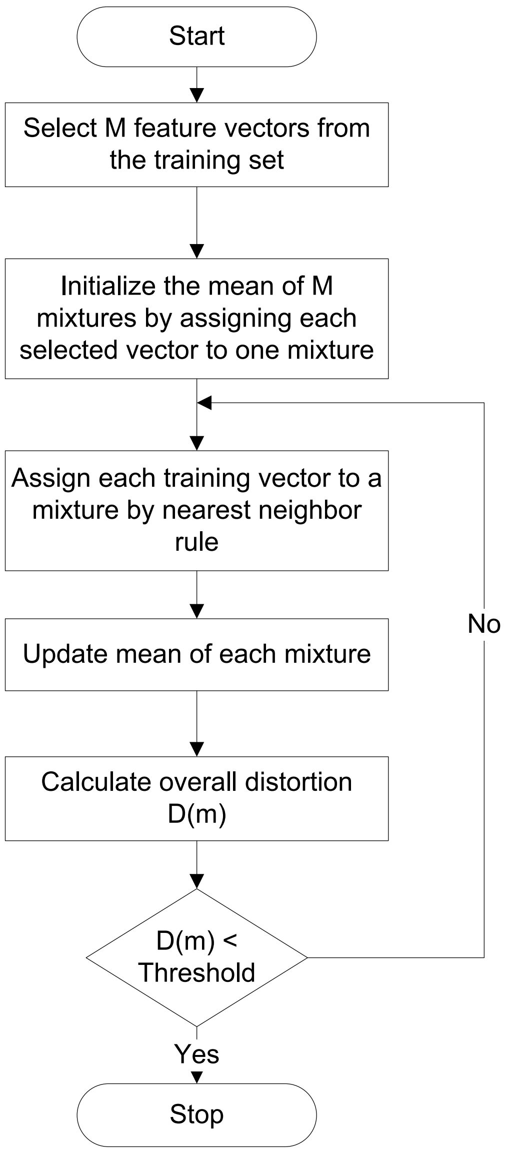
Start
Select M feature vectors from the training set
Initialize the mean of M mixtures by assigning each selected vector to one mixture
Assign each training vector to a mixture by nearest neighbor rule
Update mean of each mixture
Calculate overall distortion D(m)
D(m) < Threshold
No
Yes
Stop

Figure 3.2: The k-means algorithm

Final Model Estimation: After initial model estimation is done as explained above, let us suppose, we have an *M* mixture component initial model $\lambda^0 = \{w_i^0, \vec{\mu}_i^0, \Sigma_i^0\}$ and a set of training vectors $\{\vec{x}_1, \vec{x}_2, ..., \vec{x}_T\}$. The probabilistic count for each mixture component is defined as

$$p(i \mid \vec{x}_t, \lambda) = \frac{w_i b_i(\vec{x}_t)}{\sum_{k=1}^{M} w_k b_k(\vec{x}_t)}, \quad 1 \le i \le M \ \ and \ 1 \le t \le T \tag{3.9}$$

where, $b_i(\vec{x}_t)$ is the PDF of i^{th} component and is given by Equation 3.4. $p(i \mid \vec{x}_t, \lambda)$ can be thought of as the ratio of the probability of an observation vector belonging to i^{th} component and sum of probability of that observation vector belonging to each component. The sufficient statistics for the weights, mean and covariance matrices are calculated as given in the Equations below.

$$n_i = \sum_{t=1}^{T} p(i \mid \vec{x}_t) \tag{3.10}$$

$$E_i(\vec{x}) = \frac{1}{n_i} \sum_{t=1}^{T} p(i \mid \vec{x}_t)\vec{x}_t \tag{3.11}$$

$$E_i(\vec{x}^2) = \frac{1}{n_i} \sum_{t=1}^{T} p(i \mid \vec{x}_t)\vec{x}_t^2 \tag{3.12}$$

In the above equations, $E_i(\vec{x})$ denotes, summation of the conditional expectations of training vectors $\vec{x}_t$ for given component i. Similarly, $E_i(\vec{x}^2)$ denotes summation of the conditional expectations of square of the training vectors for given component i. Equations 3.9 to 3.12 form the first step of our training scheme which is identical to the EM Algorithm. Based on the values obtained using the above Equations and the initial model, the final model parameters are estimated in the second step as given below.

$$w_i = [\alpha_i n_i / T + (1-\alpha_i) w_i^0]\gamma \tag{3.13}$$

$$\vec{\mu}_i = \alpha_i E_i(\vec{x}) + (1-\alpha_i)\vec{\mu}_i^0 \tag{3.14}$$

$$\Sigma_i = \alpha_i E_i(\vec{x}^2) + (1-\alpha_i)(\Sigma_i^0 + \vec{\mu}_i^{02}) - \vec{\mu}_i^2 \tag{3.15}$$

Here, γ is a constant such that the summation of all weights is equal to 1. α_i is a constant known as the adaptation coefficient and is given by

$$\alpha_i = \frac{n_i}{n_i + r} \qquad 1 \le i \le M \tag{3.16}$$

In the above equation, r is a constant known as the relevance factor. It plays an important role in the adaptation process, by determining the adaptation coefficient. The adaptation coefficient, in turn, determines the extent to which a particular component is altered. A dynamic calculation of the relevance factor r is given below.

$$r = \frac{\min_i n_i + \max_i n_i}{2} \qquad 1 \le i \le M \tag{3.17}$$

The training scheme can be summarized as follows. An initial model is calculated using the iterative k-means algorithm. The BA technique, which has two steps namely expectation step and adaptations step, is used to estimate the final model parameters by adapting the initial model. In the following, we describe the second problem in GMM i.e. how to calculate the match score of a set of test vectors against a model (Data-Model Fit).

3.3.2 Estimation of Data-Model Fit

The match score of a test utterance against a word or speaker model is given by a likelihood score. This can be seen as the likelihood that the test utterance is the same as the modeled word or produced by the modeled speaker. If $\{x_1, x_2, ..., x_n\}$ is the test utterance and λ is the *M* mixture component model, then the likelihood score is given by

$$Likelihood = \prod_{t=1}^{n} \left[\sum_{i=1}^{M} w_i b_i(\vec{x}_t) \right] \tag{3.18}$$

In order to avoid any singularity in computation due to low values, usually log-likelihood score is used [25]. To account for unequal length test utterances, average log-likelihood score is preferable. The average log-likelihood score is given by

$$score = \frac{1}{n}\sum_{t=1}^{n}\log\left[\sum_{i=1}^{M} w_i b_i(\vec{x}_t)\right] \tag{3.19}$$

In this section we have explained the concept of GMM and the technique we follow to estimate the model parameters. The method used to find the data-model fit is also discussed. As mentioned earlier, to carry out GMM based experiments for our work, an evaluation platform has been developed. In the next section, we elaborate the architecture of the evaluation platform and discuss how it can be used to carry out experiments based on MFCC and GMM.

3.4 Evaluation Platform Architecture

In this section, we describe the architecture of the evaluation platform developed to facilitate and expedite the GMM and MFCC based speaker and word recognition experiments in our research work. We intend to design the architecture in a manner that apart from supporting our work, it also provides room for new techniques so that any different method of speaker or word recognition experiments can be carried out and thus making it an efficient tool for further research in this field. In order to achieve this, the evaluation platform should meet the following basic requirements.

- It should be modular and extensible and should allow the addition of new processing blocks in future.
- All the process parameters should be effortlessly modifiable by the user.
- It should facilitate the rapid creation of custom speaker recognition or word recognition configurations by selectively including and excluding blocks in the process flow.
- It should facilitate working with multiple speech databases including non-standard user databases.

The choice of the programming language for developing this platform should be compatible with the above requirements. We have used the C++ language that supports an object-oriented and modular approach. The software development has been carried out using the Borland C++ 5.0 Integrated Development Environment (IDE). The collection of objects and functions in our application bear resemblance to a list which allows addition of new functions and objects of new structures at any point of time without affecting the existing modules of the application. This collection serves as the processing blocks for the experiments and allows the user to rapidly configure the candidate methodology by selecting only the needed functions. The functions and objects involved in various units of the evaluation platform are described below.

3.4.1 System Overview

The current implementation of the evaluation platform consists of the following three high-level units.

- Speech Extraction Unit
- MFCC Feature Extraction Unit
- GMM Unit

These units are in turn broken into sub units, each corresponding to a specific function. This provides more flexibility in custom configuration of the experiments. There are various parameters whose values determine the function of these units. A speaker or word recognition experiment can be carried out by specifying the values of these parameters and using only the needed functions to configure the experiment process.

3.4.2 Speech Extraction Unit

The speech extraction unit deals with reading the speech, segmenting the speech into frames and determining whether the frames correspond to speech or silence. The path of execution in this unit is shown in Figure 3.3. The function that calculates the amplitude values of a frame operates on the output of the previous function that segments the speech into frames. However, the function which characterizes a frame as speech or silence is coded to operate on an object that contains the frame's amplitude values. This is because that there are several methods to classify a frame as speech or silence and hence a new speech frame classification method can be added into the unit as a function without affecting the existing flow of the unit.

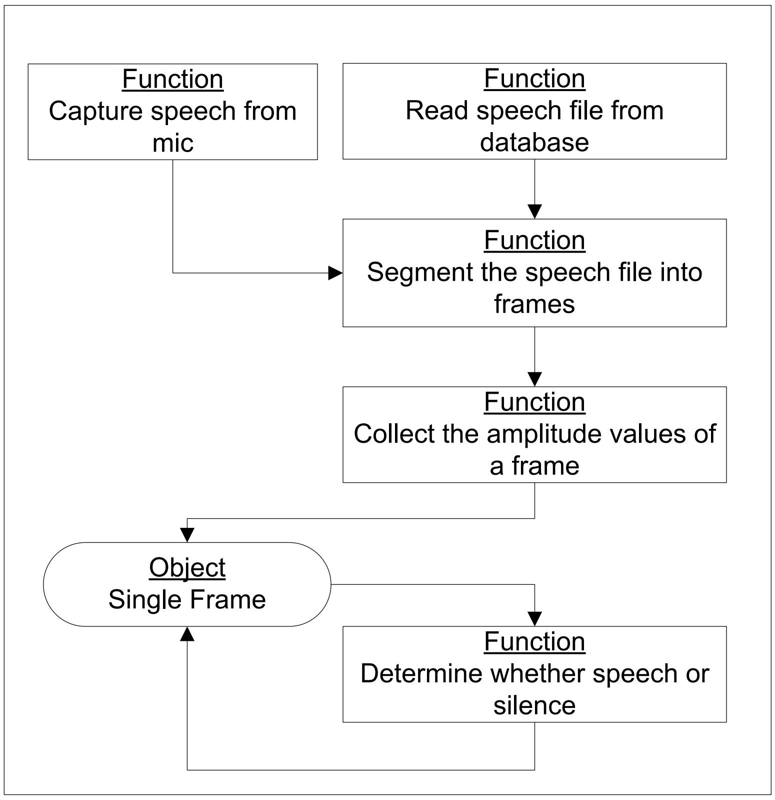
Function
Capture speech from mic
Function
Read speech file from database
Function
Segment the speech file into frames
Function
Collect the amplitude values of a frame
Object
Single Frame
Function
Determine whether speech or silence

Figure 3.3: Speech extraction unit

Working with Various Speech Databases: Existing standard speech databases differ in the organization of the speech and speaker data in it. Also, the databases use different audio formats to store the speech data and have their own naming convention to represent the speakers, recording sessions and the speech files. Hence, in order to make the evaluation platform compatible for all speech databases, a uniform structure, where each word or speaker is denoted by a unique number, has been defined. Speech databases need to be reorganized to conform to this structure. Since speech processing is better done with uncompressed speech data, all the speech files are converted into PCM wav files.

A function in the speech extraction unit, reads the data from a speech file, irrespective of its recording parameters like sampling frequency, number of bits per sample etc. The other functions in this unit segment the speech data into frames and store the amplitude values of each frame into an object.

Live Sound Capture: We have also implemented a function as part of this speech extraction unit to record speech, through a microphone connected to the computer running the evaluation platform, which can be used for both training and testing process in the experiments. This function can also be deployed to create a speech database by recording speech from various speakers as an offline step. It can record sound in mono channel PCM wav format based on the sampling rate and bits per sample values specified by the user. This function enables the application to analyze the performance of any speaker or word recognition methodology in real-life conditions. The testing process in the experiments can be carried out by considering the captured speech as input data and producing the result instantly after the speech recording is completed.

3.4.3 Feature Extraction Unit

The overview of the feature extraction unit is shown in Figure 3.4. Like the speech extraction unit, this unit is also common for both training and testing. The feature extraction unit calculates the MFCC feature vectors for a single frame. Among the sub-units in it, the Hamming window multiplication, Fourier transform and mel filter bank energy calculation sub-units take a single frame as input. It is known that after mel filter bank multiplication the output will be a vector depending on the size of the filter. Hence the remaining sub units - log energy calculation and discrete cosine transformation - deal only with single vector objects. Currently, we use a mel filter bank size of 27 and so the vector objects will be of size 27.

The function that multiplies the amplitudes of the speech frame with the Hamming window, also dynamically calculates the Hamming window values depending on the size of the frame. For mel filter bank and discrete cosine transform, we use matrices having those values stored as constants. Radix-2 decimation algorithm is used in the function that calculates the Fourier transform. Depending on the size of the input speech frame, it pads zeros at the end of the frame so that the input size becomes a power of 2.

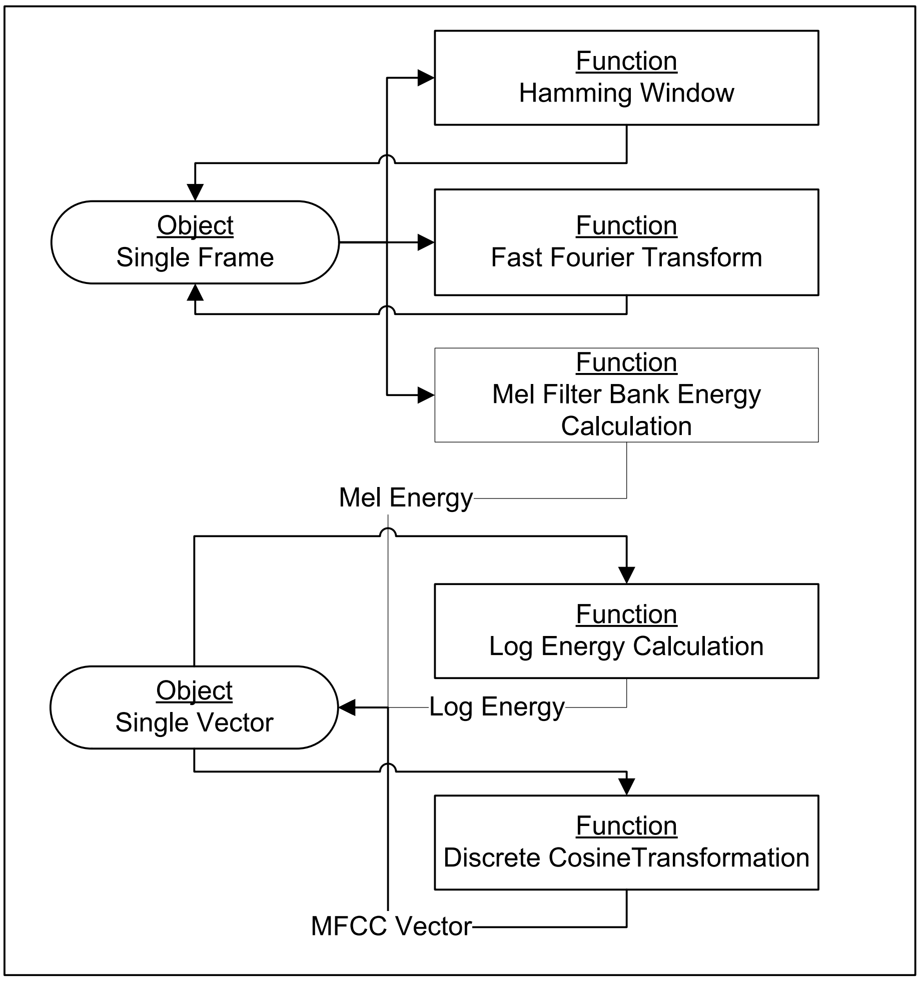

Figure 3.4: Feature extraction unit

3.4.4 GMM Unit

The three functions in the GMM unit are the initial model estimation, final model estimation and data-model fit (score calculation). The first two are used in training a model for a given collection of feature vectors and the last function is used in the testing process to find its match score against a model. The initial and final model estimation functions implemented are based on the techniques described in Section 3.3.1. The object Vector Collection shown in Figure 3.5 contains all the feature vectors to be modeled or scored. This object is in turn a collection of objects of type Single Vector shown in the feature extraction unit.

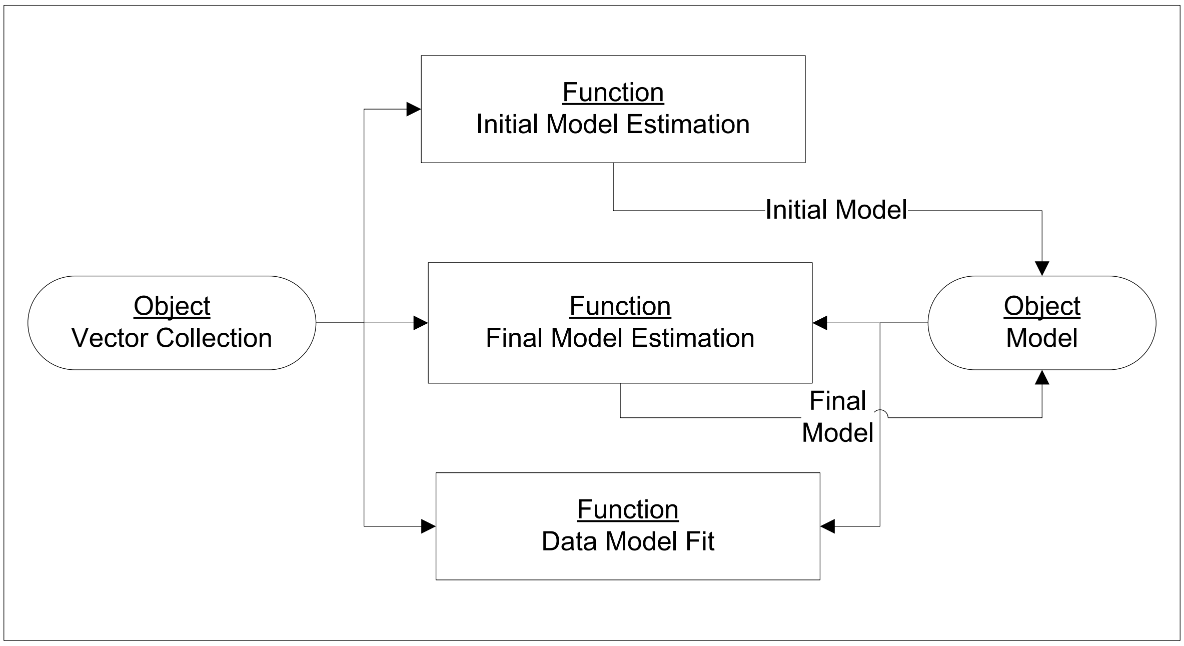

Figure 3.5: GMM unit

3.4.5 Configuration of the Experiment Methodology

In the previous three sub-sections, we described the various functions present in the three main units of the evaluation platform. As mentioned earlier, the functions not only depend on the input and output objects but also on the values of their process parameters. To run an experiment, the user should select the needed functions that form the methodology of the experiment and assign the values of different process parameters. This process is described as the Configuration Area in the Figure 3.6 that shows the overall architecture of the platform. We have developed the evaluation platform in a manner that the user can select the necessary functions and assign values to the parameters easily through a Graphical User Interface (GUI).

The selection of functions forms the basis of the experiment to be run. The functions to be included can be easily specified through the GUI. In the present form of the evaluation platform, this process can be classified into two broad categories of selections, namely, training and testing. All the three main units are involved in both training and testing and certain functions within these units are indispensable and hence the choice of selecting these functions is not provided in the platform. However, the user can consider the other functions as optional and can carry out an experiment without including them. Also, there can be different functions corresponding to the same task and hence the user should make the choice of the function to be used.

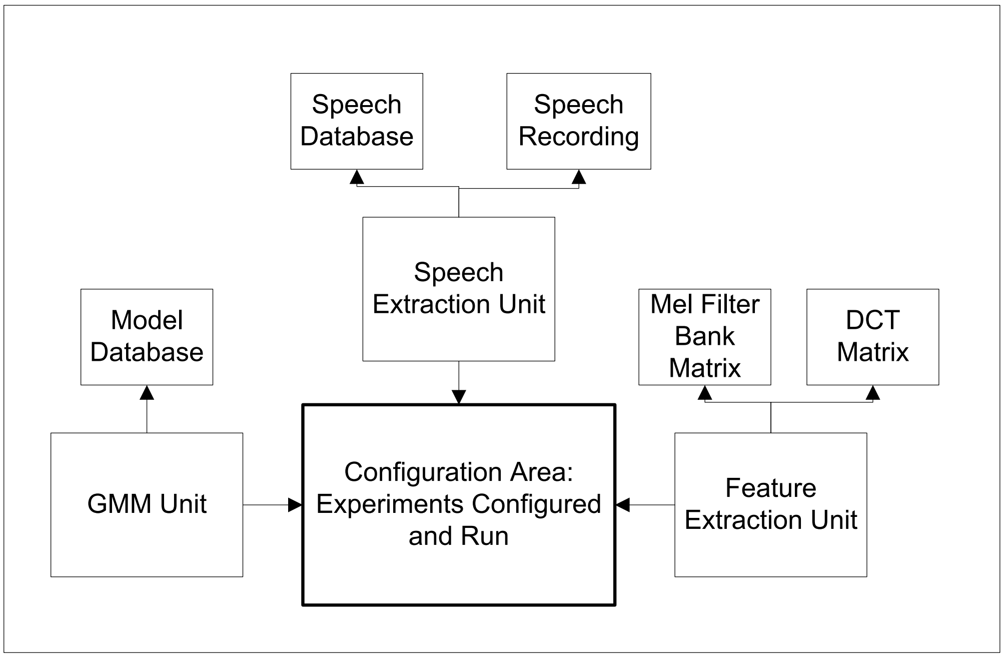

Figure 3.6: Overall architecture of the evaluation platform

In the Configuration Area of the architecture, the main program that runs the experiment is coded in a manner that it reorganizes itself according to the selections made. This program is an internal part of the platform and it is hidden from the user running it as an application. However, adding a new function to a unit requires the main program to be altered. In programming point of view, this is about adding or altering a conditional statement in the appropriate place in the main program after compiling and adding the function to the list of existing functions. This shows extensibility feature of the platform and its scope can be extended from functions to the whole unit. For instance, a LPC extraction unit can be added to provide the user with the choice to select between MFCC and LPC based feature extraction units.

To configure an experiment, the selection of the process should be accompanied by the assignment of values to various process parameters. As mentioned earlier, various combinations of these parameters should be analyzed to establish the best performance of a methodology. The various process parameters whose values are assigned by the user are listed below.

Modifiable Parameters in Speech Extraction Unit:

Database Path: This parameter denotes the path of the speech database that is used in the experiment. As described earlier, this speech database needs to be reorganized to a specific structure in order to use it.

Start and Final Speaker: The speakers in the above database are numbered from 1. These two parameters help to specify a list of speakers from the database to be considered for experiment.

Start and Final Word: The words in the above database are numbered from 1. These two parameters help to specify a list of words from the database to be considered for experiment.

Frame Width: This parameter specifies the size of the speech frame. The speech is segmented into windows of the mentioned size.

Frame Update: This parameter denotes the progressing rate of the speech window.

Modifiable Parameters in GMM Unit:

Start Co-efficient: The index of the first coefficient in a feature vector to be considered for either training or testing is specified by this parameter.

Final Co-efficient: This parameter specifies the index of the last coefficient in a feature vector to be considered for either training or testing. Together with the *Start Co-efficient* parameter, it determines the dimension of the vector.

No. Of Segments: The number of segments per word is mentioned by this parameter. The experiments involving word segmentation are explained in the following chapters. If this value is 1, it specifies a baseline GMM based experiment.

K-means Iterations: The number of iterations involved in the k-means algorithm used for initial model estimation is denoted by this parameter.

Number of Mixtures: This parameter comes into effect in the training process and it specifies the number of mixtures in the model.

Model Storing Path: This parameter specifies the location in the computer where the models are stored after training.

Model Database Path: This parameter specifies the path of the model database. This is used in the testing process for scoring.

In the feature extraction unit, the size of the vector depends on the mel filter bank size. However, it has not been kept as a modifiable parameter since the function does not dynamically calculate the mel filter bank. Instead, it uses a constant filter bank size of 27 and the mel banks are kept as constant matrices with predetermined values.

Once the experiment methodology is formed using the functions and the various parameter values, the main program run the experiment. In the case of the training process, the result will be the trained models that will be stored in the location specified by the user. The file name of a model is given a number equivalent to that of the speaker or word modeled. Along with all the models stored after the experiment, a file mentioning all the details of the training process including the value of all the parameters is generated and stored along with the model. This file denotes the conditions under which the experiments are carried out. For the testing process, the experiment will produce a text file that contains the results of the experiments. This result file, apart from stating the overall word or speaker recognition accuracy, also contains the details about various parameters and mentions the match score obtained for all the test data against all the models.

3.5 Summary

In the beginning of this chapter, we explained the MFCC feature extraction, GMM training and GMM testing processes that will be used in our research work. The GMM training procedure consists of the preparation of an initial model using 10 iterations of the k-means algorithm followed by the estimation of the final model using a variant of the Bayes Adaptation method. An evaluation platform to carry out both speaker and word recognition experiments based on GMM and MFCC was developed and its description was given in the later part of the chapter. We provided a GUI based interaction between the user and the platform so that any desired experiment methodology can be rapidly configured without the need to manually alter the contents of the program. This platform was also developed in a modular manner so that new techniques can be added at any stage without affecting the existing architecture. We envisage that this feature will make the evaluation platform a useful tool not only for our work but also for future research on speaker and word recognition using different techniques. We carried out all our experiments, explained in the following chapters, using this platform.

Chapter 4

GMM Based Speaker Identification

4.1 Introduction

It has been established from the literature survey, that GMM is the most successful and widely used technique for speaker recognition. GMM has been mostly used in the text-independent scenario, where a test utterance of 6 to 10 seconds duration is normally needed for correct speaker recognition. Hence, applying text-independent speaker recognition for our work will not be productive since we have to deal with single-word test utterances typically of about half a second duration. Since our work deals with a limited and closed vocabulary, the set of words that are allowable as test utterances are known beforehand. This knowledge can be utilized to move the speaker recognition problem from a text-independent scenario to text-dependent or text-constrained scenario. Generally, in the speaker recognition problem, by constraining the speech, better performance than text-independent speech can be obtained because the comparison is made between how speakers produce certain specific sounds rather than needing to represent and compare speakers over large textual variations [46].

A text-dependent speaker identification system uses the same word or phrase in both training and testing. A text-constrained speaker identification system uses a set of words in training and it is assumed that the test utterance will only contain one or many words from the same set. Therefore, for our work, which operates on a set of words, a text-dependent system should exactly know which word from the vocabulary is being considered for testing whereas in a text-constrained system, such information is not necessary. In the following, we evaluate text-dependent and text-constrained speaker identification approaches using GMM. The rest of the chapter is organized as follows. In

Section 4.2, we evaluate and present the results of GMM based text-constrained speaker identification. In Section 4.3, we propose a new technique for text-constrained speaker identification using sub-words and present the results. Section 4.4 provides the results obtained for GMM based text-dependent speaker identification and Section 4.5 summarizes this chapter.

4.2 Text-Constrained Speaker Identification

The text-constrained speaker identification system uses speech only from a specific acoustic group, such as words or phonemes to train the speaker model. During testing, only speech from the same group is used. The concept of text-constrained speaker identification is proposed by researchers mainly to improve the accuracy in text-independent speaker recognition system [81, 82]. It is based on the idea that by constraining the system to a particular acoustic group, the speaker identification task is made to focus only on speaker differences for that particular group to improve the accuracy. It was suggested to use a speech recognition front end to segment the unconstrained speech into acoustic units and then limit the speaker modeling and verification processes to certain preselected acoustic units. Reynolds [46] implemented GMM based text-constrained speaker verification based on the above mentioned procedure and reported that the main drawback is the poor consistency of the speech recognition front end on unconstrained speech thereby eliminating the specificity of the text-constrained models.

In our work, we form the GMM based text-constrained speaker recognition system by considering all the words in the vocabulary as one acoustic group. We eliminate the speech recognition front end in the above said approach by assuming that only those words are uttered by a speaker during training and one of those words is uttered during testing.

4.2.1 Experimental Results

Speech Database:

TI46 [83] is an isolated word database which contains two directories, TI20 and TI46. In all the experiments in our work, we consider the words from the TI20 directory. The TI20 directory contains 20 words. They are the 10 digits, *ZERO, ONE, TWO, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT* and *NINE* and 10 commands, *ENTER, ERASE, GO, HELP, NO, RUBOUT, REPEAT, STOP, START,* and *YES*. These words are uttered multiple times by 8 male and 8 female speakers. There are two sessions namely TRAIN and TEST in this directory. TRAIN session contains 10 utterances of each word by each speaker and TEST session contains 16 utterances of each word by each speaker.

Experimental Set-up:

Each of the 16 speaker models is trained by taking utterances of all the 20 words by the speaker from the TEST session. The number of utterances of each word used for training is varied to find its effect on the performance. 10 iterations of the k-means algorithm are used to initialize the model and the training scheme explained in the last chapter is used to estimate the final model. During testing, since it is closed set speaker identification, the test utterance is scored against the model of all the speakers and the model against which the highest score is obtained is considered to be the speaker. All the utterances of the words available in the TRAIN session are considered as test utterances. In total there are 3186 test utterances excluding few corrupted files in that session. For both training and testing process, 20 MFCC feature vectors are calculated from a 20 msec speech window progressing at a rate of 10 msec. This set-up of feature vector extraction is considered throughout our work.

Results and Discussion:

The results from the experiment are presented in Table 4.1. It shows that the effect of training speech is similar to that of the GMM based text-independent scenario observed in the literature survey [25]. As the number of utterances of each word used for training increases, the recognition rate also increases. Though a high recognition accuracy of

96.6% is obtained in our experiment, the factor that mainly determines the robustness of the text-constrained speaker recognition concept in general is the size of the acoustic group. If the size of the acoustic group used in the text-constrained system is large, then more speech from the group is required during testing to determine the speaker differences. Therefore in our work, if the number of words that constitute the acoustic group is more, speaker identification using a single word utterance will be difficult. This can be observed from Table 4.2 which gives the result obtained from the experiment explained in the following.

No. of Utterances of Each Word in Training	Speaker Recognition Rate (%)			
	No. of Mixtures: 8	No. of Mixtures: 16	No. of Mixtures: 24	No. of Mixtures: 32
1	83.64	86.62	89.29	89.14
2	84.93	88.23	91.08	92.34
4	88.54	91.39	94.03	95.51
6	89.54	92.37	94.76	96.45
8	88.39	92.75	95.29	96.54
10	88.54	93.03	94.98	96.60

Table 4.1: Text-constrained speaker recognition

No. of Words in the Acoustic Group	Speaker Recognition Rate (%)			
	No. of Mixtures: 8	No. of Mixtures: 16	No. of Mixtures: 24	No. of Mixtures: 32
10	90.71	93.66	97.05	97.55
5	95.98	97.86	98.49	98.74

Table 4.2: Text constrained speaker recognition with different words-group size

Effect of the number of words in the acoustic group: We examined two different text-constrained systems with the same training and testing set-up used in the previous experiment. The first system is limited to the 10 digits. The second system is constrained to the 5 words, *ENTER, ERASE, GO, HELP* and *NO*. As a result of the finding in the previous experiment, we used 10 utterances of each word in the training. The speaker recognition rate obtained for the two systems are listed in Table 4.2.

Observation of the results in Table 4.1 and Table 4.2 clearly indicates that increasing the number of words in the text-constrained system results in reduction of accuracy. The drop in the recognition rate is 2.14% when the number of words is increased from 5 to 20. It is known that different types of sounds are present in each of the words in the group and hence there will be common or similar sounds across the word group. In the following, we examine whether this information can be utilized to improve the performance of text-constrained speaker identification when more number of words is used. We propose and evaluate a concept of making the text-constrained speaker identification work on similar sub-word units present in the group.

4.3 Sub-Word Constrained Speaker Identification

To improve the robustness of text-independent speaker recognition, researchers have tried comparing only the similar sounds produced by different speakers. In [84], a phonemic decoder is used to associate each speech feature vector with a corresponding phoneme and based on this information, the feature vectors are grouped into broad phone-classes such as vowels, fricatives, and plosives. Subsequently, a separate GMM is estimated for each phone-class and testing is done by scoring the sounds in the test utterances only with the group that it belongs to. We apply such a grouping of similar sounds in the text-constrained system but without the above mentioned phoneme decoder module which typically rely on transcriptions of the speech utterances. In [44], that presented a speaker recognition system based on HMMs of sub-word units, it was shown that there is only a small difference in performance between sub-word units formed with and without phonetic transcriptions of the utterances. Hence we are inclined towards a setup in which the utterances are segmented into sub-word units without using any linguistic knowledge.

Since such sub-word units do not explicitly carry any phonetic information, a new scheme has to be formulated to group similar sub-words. In the following, we explain the segmentation approach taken to form the sub-words and propose a new scheme to group similar sub-words.

4.3.1 Segmentation Methods

As mentioned earlier, we adopt a segmentation technique that is not based on any linguistic information. The simple and straightforward approach is to segment a word into equal segments. This makes the boundaries of each of the segment within a word fixed and hence the length of the segments for different words or for different utterances of the same word will be different. Another approach is to use overlapping segments and make the length of the sub-word segments across the system same. The two approaches are briefly explained in the following.

Equal Segments: The number of sub-word segments per word is predefined and is the same for all the words in the system. Each word is divided equally into that number of segments and if the speech feature vectors representing a word cannot be divided evenly among the segments, the last segment will contain more vectors than the rest of the segments.

Overlapping Segments: The length the of sub-word segments for all the words in the system is predefined in this method. Each word is divided into variable number of overlapping segments of predefined fixed length. The length is expressed in terms of the number of speech feature vectors.

4.3.2 Sub-Word Grouping Approaches

The technique proposed by us to group similar sub-words is based on the GMM matching process. To classify the sub-words into C classes, we first define a classifier GMM with C mixture components. This GMM is trained by taking utterances of all the words in the system uttered by all the speakers. This GMM is intended to group all the speech feature vectors in all the words into C classes. The k-means clustering technique, which is applied for initial model training, assigns each speech feature vector to the cluster with the nearest mean. Hence, it is assumed that each mixture in the model contains similar speech vectors and represents one sound class. As explained in the previous chapter, the final model estimation in GMM is derived from these initial model parameters.

For modeling a speaker, each of the sub-word segments of the words uttered by the speaker during training, have to be first associated with one of the C classes. This is done by scoring all the vectors in the segment with each of the C mixtures in the Classifier GMM and finding which mixture has the highest cumulative score. That is, for a segment, the match score of each of its speech feature vector against a class is calculated to find the overall match score of the segment against the class. After this process, all the sub-words in the training speech are grouped into C classes and a GMM is trained for each of the C classes using all the vectors belonging to the segments associated with that class. Hence, each speaker will have C sub-models, each representing a sound class.

During testing, each segment in the test utterance is associated with a sound class. This is carried out using the classifier GMM as done in the modeling stage. After this process, the segment is subjected to scoring only against the sub-model corresponding to that class. The overall matching score of a word against a speaker's model is obtained by the sum of the scores obtained by all the segments in that word. The parameters that have a significant impact in this sub-word constrained speaker identification system are the number of sound classes used and the number of segments per word. In the following, we describe the experimental results of this system obtained with different combination of these two parameters.

4.3.3 Experimental Results

Experimental Set-up:

Experiments are carried out in the TI46 database. For the MFCC set-up and the GMM training, the procedure used in the previous experiment is followed. All the 20 words used in the previous experiment are used here. For speaker model training, TEST session of the speech database is used and for speaker identification process, 3186 utterances from the TRAIN session of the speech database are used. The classifier GMM is prepared by taking one utterance of each word uttered by all the 16 speakers. 10 utterances of each word are used for speaker model training. Different values (8, 16, 24 and 32) are examined for the number of mixtures in each sub-model of the speakers. The two types of segmentation methods explained earlier are carried out. When the words are segmented as overlapping segments, the overlapping length is kept at half of the length of the segment.

Experimental Results:

The results obtained when words are segmented using the equal segmentation method and the overlapping segmentation method are presented in Table 4.3 and Table 4.4 respectively. The recognition rate shown in the tables are the best result obtained from the examination of the different values of mixtures in the sub-model. It can be observed from these tables, that the highest recognition accuracy obtained is almost identical to that of the text-constrained system presented in Table 4.1. The effect of sound classes is examined by keeping the number of classes at 2, 4 and 6. From results, it can be observed that better performance is obtained when the number of classes is 2. When the number of classes is increased, the performance decreases significantly. However, it has to be noted that keeping the number of sound classes at 2 is too coarse-grained and may result in grouping of sub-word segments with significantly different acoustic characteristics together. The results indicate the need for more robust grouping approaches to classify the sub-words into several sound classes.

Analysis of the effect of the segmentation method in sub-word constrained speaker recognition system shows that as the number of segments per word increases the accuracy also increases. In Table 4.3, when the number of segments is increased from 2 to 6, the recognition rate increases irrespective of the number of sound classes used. Similarly in Table 4.4, the increase in accuracy is observed when the length of the segments is less corresponding to more segments per word. It can perhaps be inferred from the above that small segments are associated with distinct elementary sounds and hence lend well for being grouped into sound classes that are distinguishable from each other.

No. of Segments per Word	Speaker Recognition Rate (%)		
	No. of Classes: 2	No. of Classes: 4	No. of Classes: 6
2	93.11	84.61	78.02
3	94.87	89.06	83.41
4	95.82	92.74	87.48
5	95.63	94.29	91.58
6	96.61	95.61	92.45

Table 4.3: Sub-word constrained speaker recognition with equal segmentation

Length of the Sub-Word Segment	Speaker Recognition Rate (%)		
	No. of Classes: 2	No. of Classes: 4	No. of Classes: 6
10	96.62	96.03	93.34
12	96.34	95.95	92.01
14	96.32	95.89	90.85
16	95.65	92.84	87.45
18	95.02	90.44	85.27

Table 4.4: Sub-word constrained speaker recognition with overlapping segmentation

4.4 Text-Dependent Speaker Identification

Text-dependent speaker identification systems seek to associate an unknown speaker with a member from a registered population, provided the phrase uttered by the speaker is known beforehand to the systems [5]. In general, among all types of speaker recognition systems, text-dependent systems are more accurate, since both the content and voice can be compared. Text-dependent scenario is generally applied for speaker authentication systems for applications such as access control, where a fixed phrase is used as password and the registered speaker models are trained using only the utterances of that phrase. But when such systems have to deal with a situation where the input phrase can be one of many permitted phrases, the number of speaker models per speaker will be more depending upon the set of phrases permitted in the system. In that case, for each phrase in the set, all the speakers should be trained using utterances of that phrase. So, if there are S speakers and N phrases in the system, then in total, there will be $S \times N$ speaker models. However, it has to be noted that even in this case, the phrase uttered has to be made known to the system so that it considers only the speaker models trained with that phrase for speaker recognition.

In the following, we evaluate GMM based text-dependent speaker recognition using the set of 20 words and 16 speakers from the TI20 directory of the TI46 database. Determining the optimal number of mixture components and analyzing the robustness of the model for different number of training utterances are the main focus of the experiments. Also, the words in the vocabulary are distinguishable from each other in terms of size and type of sounds in it. Hence the performance of text-dependent speaker recognition for each of the words is examined separately.

4.4.1 Experimental Results

Experimental Set-up:

Like the previous experiments mentioned in this chapter, the TEST session of the database is used for training the speaker models and 20 MFCC feature vectors are calculated from a 20 msec speech window progressing at a rate of 10 msec. For each of the 16 speakers, 20 speaker models are prepared corresponding to the 20 words in the database. We refer to these models as speaker-word models. Each speaker-word model is prepared by taking multiple utterances of that particular word uttered by the speaker. The initial and final model training are done similar to that of the previous experiments. The number of mixture components in the GMM is varied to find the optimal value. The

TRAIN session is used for testing and there are a total of 3186 test utterances. For each test word, the speaker models trained using only that word is considered for speaker identification.

Experimental Results:

The overall speaker recognition rate obtained for all the 3186 test utterances is listed below in Table 4.5. The results show that speaker recognition rate increases when the number of training utterances is increased. The best result obtained from the experiment is 99.28% when speaker-word models are trained using 10 utterances of the word. It can also be observed that the best recognition rates obtained are high and almost similar when the number of training utterances per model is varied from 7 to 10. In all the cases, the optimal number of mixtures is found to be 8 even though the number of training vectors is diverse due to the different number of training utterances used.

No. Of Training Utterances	Speaker Recognition Rate (%)			
	No. of Mixtures: 4	No. of Mixtures: 8	No. of Mixtures: 12	No. of Mixtures: 16
5	98.02	97.93	97.08	96.73
6	98.43	98.68	97.96	97.80
7	98.52	99.00	98.61	98.61
8	98.68	99.02	98.96	98.80
9	98.74	99.09	99.05	98.90
10	98.93	99.28	99.24	99.02

Table 4.5: Text-dependent speaker recognition

Table 4.6 presents the speaker recognition rate obtained separately for each of the 20 words, when the number of training utterances is 10 per model and the number of mixtures is 8. There are 160 utterances for each word uttered by 16 speakers (a few words contains 1 or 2 utterances less than 160). The results show that 8 words produced 100%

recognition rate and none of the words give a recognition rate far below the overall rate of 99.28%. This indicates that GMM based text-dependent speaker recognition is capable of achieving consistently high recognition accuracy for a wide range of words.

Test Word	Speaker Recognition Rate (%)
ZERO	98.74
ONE	100.00
TWO	99.38
THREE	98.12
FOUR	100.00
FIVE	97.48
SIX	99.37
SEVEN	100.00
EIGHT	99.38
NINE	100.00
ENTER	100.00
ERASE	100.00
GO	100.00
HELP	100.00
NO	99.37
RUBOUT	99.38
REPEAT	99.37
STOP	98.75
START	96.86
YES	99.37

Table 4.6: Text-dependent speaker recognition rate obtained for each word utterance

4.5 Summary

In this chapter, we examined three different approaches for GMM based speaker identification. In the text-constrained speaker identification system, we considered the closed set of words used in the system as one acoustic group. Experiments are carried out using 20 words from TI46 database and each speaker model is trained using the isolated utterances of each of the words. The experiments revealed that training the speaker models with one utterance of each word, corresponding to approximately 10 seconds of training speech, yields recognition accuracy close to 90%. It was observed that increasing the amount of training speech increases the recognition accuracy. When 10 utterances of each word are used in training the accuracy reaches 96.6%.

We examined the sub-word based text-constrained speaker identification system based on the concept of comparing only similar sounds to determine the speaker differences. Each word is segmented into acoustic segments and we proposed a GMM based frame wise scoring against a classifier GMM to group similar segments. Experiments are carried out with two different segmentation methods, the equal segmentation of the word into a definite number of segments and the segmentation of the word into segments of fixed length. The best recognition rate achieved by both the methods is almost identical to the 96.6% obtained using text-constrained speaker identification.

The text-dependent speaker recognition approach produced an overall recognition accuracy of 99.28% for the 20 words set with 8 of the 20 words giving 100% accuracy. Out of the three techniques examined, the text-constrained and text-dependent techniques are computationally simple. However, unlike the text-constrained approach, text-dependent speaker recognition requires knowledge about the word uttered in order to identify the speaker. In line with our goal of building an integrated speaker and word recognition system, we present our work on developing a GMM-based solution for word recognition in the next chapter.

Chapter 5

Adapting GMM for Isolated Word Recognition

5.1 Introduction

Word recognition systems have normally relied on techniques that utilize the time sequence information of the speech and characterize it in the model that represents a word. The time sequence information of the speech plays a crucial role in recognizing a word utterance. Existing techniques such as the template model based Dynamic Time Warping (DTW) and the stochastic model based Hidden Markov Model (HMM) use temporal information for word recognition. However, the high computing power required by these techniques will be a major drawback when it comes to implementing a word recognition process in a device with limited resources. Hence, it is worth investigating other techniques as well, which are not often used for solving the word recognition problem.

GMM is a widely used technique in speech-related research. It has been successfully used in the field of speaker recognition. Also, in HMM, the states within are normally represented as GMM. The advantage GMM possesses is that it has relatively less computational complexity compared to the above mentioned popular techniques used for word recognition. However, while using GMM for word recognition, this advantage is suppressed by the drawback that it fails to capture the time sequence information of speech. It is known that GMM follows a bag-of-frames approach to model the statistical distribution of all the feature vectors representing the speech frames. This causes the loss of the temporal information of speech that can be obtained from the time ordered sequence of the feature vectors. Because of this drawback, there has been no major work done to use GMM for word recognition problem. Hence, in order to use GMM for word recognition, a methodology that is different from conventional GMM based modeling should be devised. This methodology should enable the modeled word to retain the time sequence information of the speech in some form.

In this chapter, we propose a technique that uses a GMM based sub-word level modeling rather than modeling the whole word using GMM. The time sequence information is preserved in the order of the GMM sequence which represents the order of the sub words that form the word. This enables us to have a sub-word level testing which can be seen as a method of matching the temporal sequence of sub-words of a test utterance with that of a modeled word. We explain this technique in detail in Section 5.3 and also discuss about the segmentation algorithms used to form the sub-words. The results obtained using the proposed technique are presented in Section 5.4. Before we explain the proposed technique, we present the results obtained using the baseline GMM approach in Section 5.2. In Section 5.5, we present an extension of the proposed technique, where multiple models are used to represent a word. Section 5.6 summarizes this chapter.

5.2 Baseline GMM Approach for Isolated Word Recognition

This section explains the drawback in using the baseline GMM approach for isolated word recognition. The word recognition result obtained using this baseline approach is also presented here. This result is used as a benchmark to examine the performance of our proposed technique which is explained in the next section. In order to model a word using GMM, the first step is to extract the feature vectors from each of the speech frames available in the word. Then, for training, the GMM technique, as explained elaborately in Chapter 3, takes these feature vectors as input and produces a model that maximizes the likelihood of the feature vectors for that model. It can be observed that the sequence of the feature vectors is of no importance in this modeling process and the model does not retain any information about the order in which they appear. This means there is a vital loss of information related to the sequence of the speech units that constitute the word.

The above mentioned drawback in GMM makes it necessary to look into new methods other than the conventional approach to incorporate time sequence information of the speech for word recognition. However, to study the performance of the new methods, we need the performance of a word recognition system based on baseline GMM. This can be set as a benchmark to examine the improvement in any different approach of GMM based word recognition. In the following we present the experiment results obtained using the baseline GMM method.

5.2.1 Experimental Set-up and Results

Experimental Set-up:

The TI20 directory of the TI46 database described in the previous chapter is used for all the experiments reported in this chapter as well. Each word model is trained by taking one utterance of that particular word by each speaker from the TRAIN session. Hence the training set contains 16 utterances of the word and is gender balanced. 10 iterations of k-means algorithm are used to initialize the model and the training scheme explained in Chapter 3 is used to estimate the final model. During testing, it is assumed that the test utterance will be one of the words in the training set and hence the test utterance is scored against the model of all the words and the model against which the highest score is obtained is considered to be the uttered word. All the utterances of the words available in the TEST session are considered as test utterances. In total there are 5082 test utterances excluding few corrupted files in that session. For both training and testing process, 20 MFCC feature vectors are calculated from a 20 msec speech window progressing at a rate of 10 msec.

Experimental Results:

The results from the experiments are presented in Table 5.1. The highest word recognition rate obtained is 90.6% when the number of mixtures in the model is kept at 16. The optimal number of mixture components for a model depends on the amount of training speech. Here each utterance of a word is estimated to have an approximate length of 0.5 seconds and hence the total amount of training speech duration in our experiment is about 8 seconds. As shown in the table, various sizes of mixture components for the model are tried and it is observed that initially the word recognition rate increases when the number of mixtures is increased and after reaching the peak for 16 mixtures, it again drops when higher number of mixtures is used.

No. of Utterances used for Training each Word Model	Approximate Duration of Training Speech for each Word Model	No. of Components in GMM	Word Recognition Rate (%)
16	8 seconds	32	86.46
		24	88.15
		16	90.60
		8	90.55
		4	86.88

Table 5.1: Isolated word recognition results using baseline GMM method

In the next section, we describe our proposed technique that will incorporate time sequence information of speech to improve the GMM based word recognition further. The result obtained in the above experiment will be used as a point of reference to study the increase in performance of the technique.

5.3 Proposed Technique to Incorporate Time Sequence Information for GMM based Word Recognition

Each word uttered has some sub-word units like phonemes. When a model for a word is trained using the GMM process, multiple utterances of that word are considered. The time order or the sequence of the sub-words that constitute a word will be the same for all the utterances of that word. Our proposed technique tries to utilize this knowledge to bring in the time sequence information in word modeling. We represent a word as a time ordered sequence of GMMs where the GMM sequence corresponds to the sequence of sub-word units in the word. This enables us to have a sub-word level testing which will match the temporal order of the sub-word units in the test utterance with that of the modeled word. In the following subsections, we explain about the segments (sub-words) and discuss the segmentation methods to segment a word. We also explain the training and the testing procedure followed by the proposed technique.

5.3.1 Segments

The term segment in our work refers to a sub-word unit of a word. A segment is represented by feature vectors of the frames within it. The work reported in [35] employs a similar segment based approach and prepares a GMM based segmental mixture model for text-independent speaker verification. In that work, the time sequence information is preserved by retaining the order of the speech feature vectors within each segment. Subsequently, a segmental mixture model is trained using a collection of segments instead of using the collection of all feature vectors as in normal GMM technique. However, this is unlikely to be effective in our work on word recognition, where it is beneficial to maintain the order of sub-word units rather than the order of the feature vectors within each of them.

It has to be noted that the time ordered segment sequence of a word used in our work is a non-overlapping one. The segments of a word are obtained from the sequence of feature vectors extracted for the entire word. Therefore, each segment of a word will be formed by portions in the feature vectors sequence of the word. The flow in forming the segments from the word utterance is illustrated in Figure 5.1. In the next section, we briefly discuss about the segmentation methods and explain the methods we chose for our work.

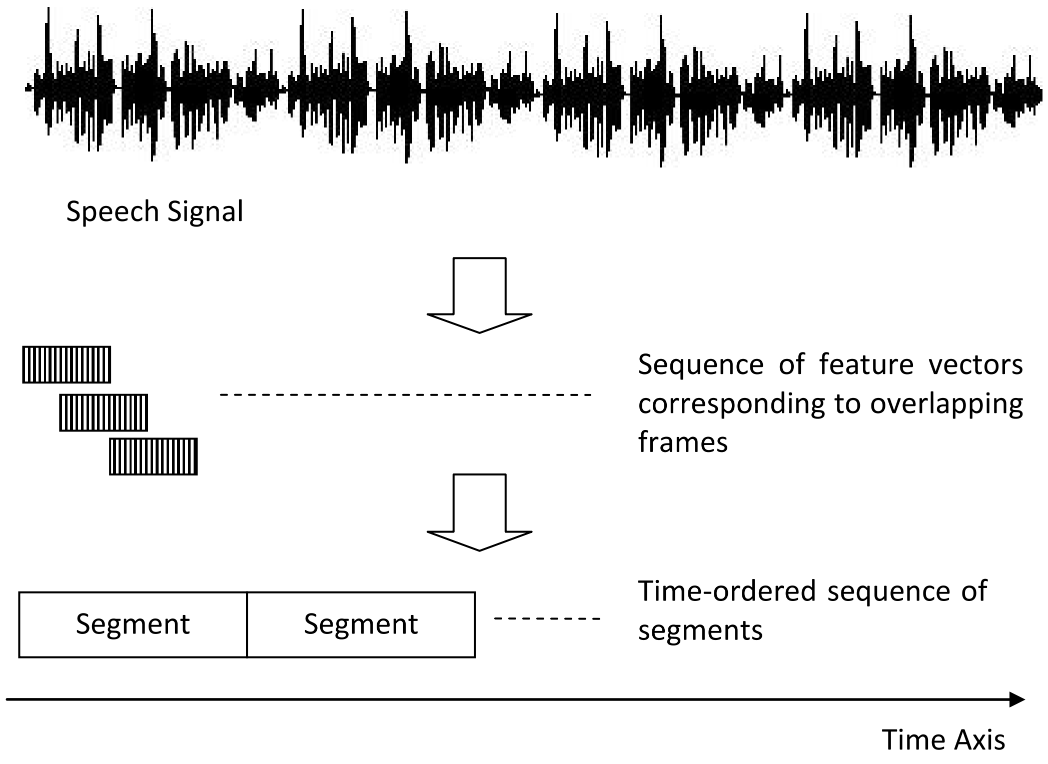
Speech Signal
Sequence of feature vectors corresponding to overlapping frames
Segment
Segment
Time-ordered sequence of segments
Time Axis

Figure 5.1: Time-ordered sequence of segments

5.3.2 A Discussion on Segmentation Methods

A vast majority of the currently available speech recognition systems for medium to large vocabulary are constructed based on sub-word units [3, 58]. These systems use the process of segmenting the speech data by optimally locating the segment boundaries and labeling those segments. The labeling procedures use linguistic knowledge such as phonetic strings. Since in our work, we are not concerned about the labeling of the segments, we focus only on the speech segmentation techniques. The algorithms found in the literature to solve the speech segmentation problem are mainly divided into two categories, hierarchical and non-hierarchical. The hierarchical speech segmentation procedures involve a multi-level, fine-to-coarse, segmentation description and use a path finding algorithm to get the segments [85]. The non-hierarchical speech segmentation procedure uses various techniques to locate the sub-word boundaries. Works reported in literature show that non-hierarchical segmentation can be done by minimizing a distortion metric using dynamic programming based methods [86] or by maximizing the score metric of acoustic models [87, 88]. However, the methods used in all the above said techniques, like the path finding algorithms and the dynamic programming, make the segmentation process compute-intensive.

The problem of automatic speech segmentation can also be classified into two different scenarios depending on the information available prior to segmentation. In the first kind, the linguistic knowledge of the speech such as phonetic transcription or the number of phonemes present in it is known and in the second kind, no linguistic knowledge about the given speech is available. If the linguistic knowledge of the speech is known, then the segmentation algorithm is required only to optimally locate the sub word segment boundaries. The above mentioned approaches for segmentation are used in this case and these approaches are typically computationally complex. Some segmentation algorithms consider the syllable-like units as the fundamental parts of speech. An algorithm for automatic segmentation of speech into syllabic units has been proposed by Mermelstein [89]. This algorithm is based on using a loudness minimum as an indicator of syllabic boundaries. In a similar work, the segmentation of speech into syllabic units is achieved based on locating local minima in the waveform amplitude [90]. The drawbacks in these algorithms are many. First of all, there is no universal agreement on a rigorous definition of a syllable [91]. Also these methods will frequently detect more candidates for a syllable boundary than are required and hence it has to be checked with the information about the number of syllables in each word. Furthermore, these algorithms are not shown to be fully accurate in spotting the syllables. These shortcomings will bring a negative effect in our proposed technique if we use the linguistic based segmentation algorithms described above. In our technique, as described earlier, the sequence of sub-words plays a major role. Hence if there is any miss in detecting a sub-word unit by these algorithms as mentioned above, the main concept of preserving the time sequence information using the order of sub-words will be affected. Because of these drawbacks in linguistic based segmentation, detecting sub words without any information about the linguistic knowledge of the word is looked at. Also, the proposed technique does not necessarily need to have any linguistic elements as segments and it can operate purely at the acoustic level.

If there is no linguistic knowledge about the given speech data, the segmentation algorithm determines the optimal number of sub-word units present in the word, as well as their boundary locations, based only on the acoustic data. As mentioned above, for our work, the number of the segments that has to be detected from various utterances of a same word should be the same. Hence, we decide the number of segments first and then use the segmentation algorithm to find the optimal segment boundaries. In the following, we explain two techniques to segment a word into a predetermined number of segments. These two techniques are used in the experimentation of our proposed technique.

5.3.3 Simple Equal Segmentation

The simplest and most straightforward way of segmenting a given word into n segments is to divide it into n equal parts. This method will make the corresponding segments in various utterances of a word to contain almost the same portion of the word if there is no significant variation in all the utterances of the word. This segmentation process is explained as follows.

Let us assume that a word is represented by a sequence of feature vectors of length v. If the number of segments desired is n, then the segmentation is carried out depending on one of the following two criteria.

- One criteria is $v = nm$, for some natural number m. That is the number of feature vectors is a multiple of n, the number of segments. In this case, every consecutive m feature vectors are assigned into different segments. So, each and every segment contains equal number of feature vectors.
- Another criteria is $v \neq nm$, for any natural number m. That is the number of feature vectors is not a multiple of n. So here it is considered that $v = nm+k$ where k = 0,1,2.. n-1. In this case, every consecutive m feature vectors are assigned into the segments numbered from 1 to n-1, while the last segment contains $m+k$ feature vectors.

5.3.4 Constrained Clustering Segmentation

The second segmentation technique used in our experiments is an adaptation of the segmentation method proposed in [86]. This technique clusters the input utterance into consecutive non-overlapping clusters and this equals a vector quantization codebook design, subject to the constraint that all the vectors contained in a cluster are contiguous in time. The technique finds the segment boundaries by minimizing the total distortion for a specific distortion measure between segments. A distortion measure based on likelihood ratio has been proposed in [86]. However, we make use of a simpler distortion measure, namely the Euclidean distance.

Let us assume that a word of *T* frames has to be segmented into *n* segments. The set of segment boundaries, t_b, b=1,2...n, is obtained such that the global distortion measure given in Equation 5.1 is minimized.

$$D=\sum_{b=1}^{n}\sum_{t=t_b}^{t_{b+1}-1} d(y_t,c_b) \tag{5.1}$$

In the above equation, y_t is the feature vector for the tth frame, c_b is the centroid of the bth segment and *d* is the local distortion. The centroid of a segment is computed as the mean of the vectors in that segment.

Among the two techniques to segment a word described above, the simple equal segmentation technique involves negligible computation. Whereas, the second technique of constrained clustering segmentation is a clustering process that involves many iterations of computation and has a relatively high computational complexity. In our experiments, both the techniques are used to segment the word utterance and the results are compared. The training and testing procedure using the segments are described in the following.

5.3.5 Training Procedure

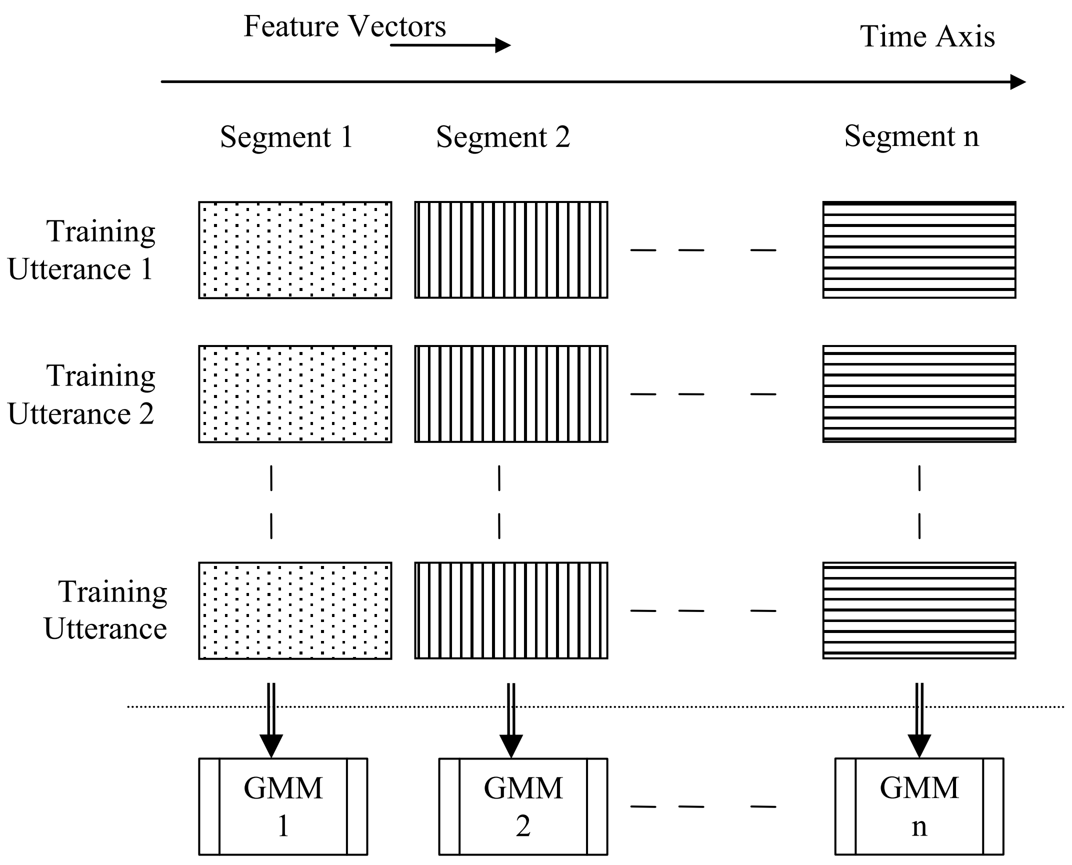

Figure 5.2: Training process in the sub-word constrained GMM technique

The aim of the training process is to model a word as a time ordered sequence of GMMs where each GMM represents a segment (sub-word) of that word and the sequence of GMMs corresponds to the sequence of the sub-words present in that word. Each of the many utterances of a word that are used to train that particular word is segmented into a definite number of segments. This amounts to transforming each utterance into a sequence of sub-words. The modeling of a sub-word can be achieved by considering only the corresponding segments from all the utterances of the word which are represented by a set of feature vectors. The training process is illustrated in Figure 5.2. The representation of a word with multiple GMMs in our technique may resemble a left-to-right continuous HMM. However, it has to be noted that there is a substantial difference between our technique and HMM. In HMM, the states that are modeled as GMM do not directly represent any specific sub-word portion. Also, the starting state and the movement from one state to another are defined by the initial state probabilities and state transition probabilities respectively. Our technique is simple where the GMMs in the word model are obtained by training specific sub-word segments. Since the sequence of sub-word segments of a word is obtained by the segmentation method, there is no need to define the transitions between the models in the GMM sequence.

5.3.6 Testing Procedure

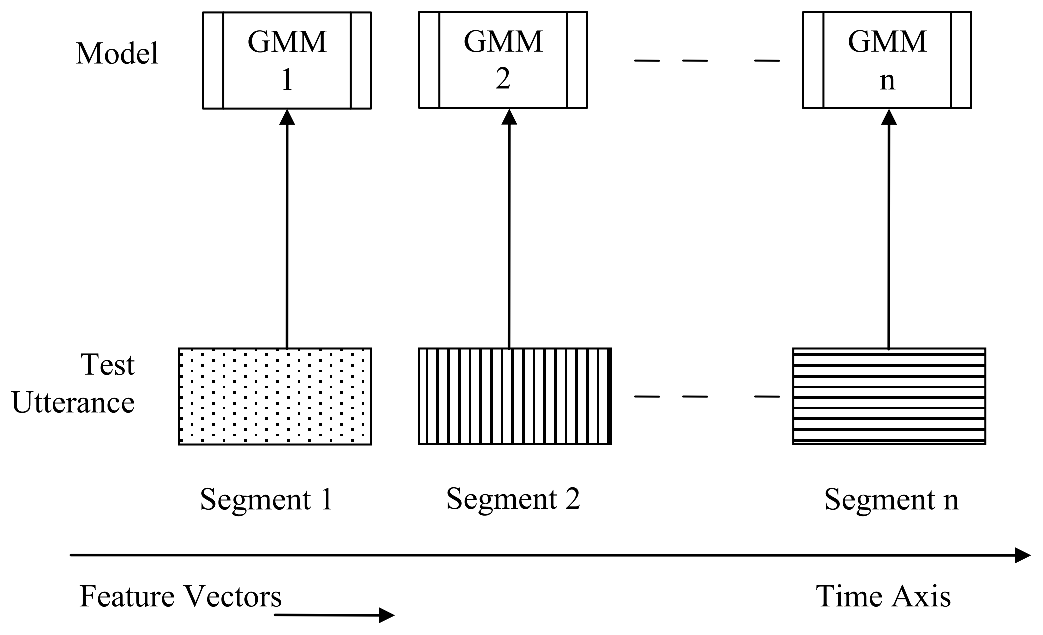

Figure 5.3: Testing process in the sub-word constrained GMM technique

In the testing process, when a test utterance is scored against a word model represented as a GMM sequence, we intend to perform the GMM based matching between the temporal order of the sub-words present in the test utterance and that of the modeled word. We segment the test utterance and form a sequence of segments, each represented by certain number of feature vectors. The segments are matched with the corresponding GMM of the word model. The feature vectors of the first segment are scored only against the first GMM in the model; those of the second segment are scored only against the second GMM and so on. The overall matching score is obtained by aggregating the matching score obtained for all the segments. It has to be noted that in order to achieve this scoring process, the segmentation technique and the number of segments used should be the same in both training and testing process. Figure 5.3 illustrates this testing process.

5.4 Experimental Results

Experimental Set-up:

The experiments are carried out using the TI20 directory of the TI46 database described earlier in this chapter. The feature vector extraction process, number of utterances used to train a word, the model initialization and the final model estimation process are kept similar to the experimental set-up for the isolated word recognition using baseline GMM approach explained in Section 5.2. Both the segmentation techniques explained earlier are employed to segment the word. The number of segments per word is kept same for all the 20 words used in the experiment. The number of mixtures used in the model for each of the segments is also uniformly maintained. The number of mixtures is kept as 4 for this experiment. As explained earlier, a total of 5082 utterances are used for testing.

Experimental Results and Discussion:

Segmentation Method	Word Recognition Rate (%)			
	Segments/ Word=2	Segments/ Word=3	Segments/ Word=4	Segments/ Word=5
Simple Equal Segmentation	95.28	95.69	96.26	95.49
Constrained Clustering Segmentation	89.61	94.10	95.06	93.94

Table 5.2: Isolated word recognition results using sub-word constrained GMM

Table 5.2 shows the results obtained from the word recognition experiments carried out using the technique proposed. It can clearly be seen that all the results, except one where the number of segments per word is kept at 2 and clustering segmentation is done, significantly outperform the best result of 90.6% obtained for the baseline GMM approach. The highest word recognition rate obtained is 96.26% which is an increase of 5.66%.

It can be observed from the results that the performance of our proposed system is consistently better with the simple equal segmentation technique than with the constrained clustering segmentation, although the latter involves more logical reasoning than the first one as it determines the segments based on the variation in spectral distortions. One likely reason for the relatively low accuracy of the constrained clustering segmentation method is that it can sometimes result in segments with a very small number of feature vectors that are insufficient for estimating a reliable GMM. From the above, it is evident that the technique of sub-word constrained GMM for word recognition is feasible with no major additional computations that are normally needed for word segmentation.

Segment-Mixture Relationship: The results presented in Table 5.2 also show that in the case of the simple equal segmentation, the recognition rate appears to be largely independent of the number of segments per word. However it has to be noted that the results shown are obtained by keeping the number of mixtures at 4 for all the cases. The word recognition accuracies obtained when the number of mixtures is varied is presented in Figures 5.4 to 5.7. It can be observed from these figures that though the accuracy varies only slightly for number of mixtures ranging from 2 to 8 in all the cases, it is on an attenuating path for higher mixture sizes. It is because when the number of mixtures is increased, the amount of training data i.e. speech feature vectors per mixture will become less and this may not be sufficient enough to characterize the speech's distribution. It is known that the words are short in duration and so each segment will be having a limited number of speech features in it. Hence using a small number of mixtures is most constructive and the accuracy is largely consistent for smaller number of mixtures as seen from the figures. Also, increasing the number of segments per word will result in less training data per segment which in turn invokes the above said disadvantage of insufficient speech data per mixture even for a smaller number of mixtures. In our experiments, increasing the number of segments per word to 5 does not better the results obtained when 4 segments per word is used. It can be concluded that keeping the number of segments and mixture size within 4 will be optimal for this technique of word recognition. The results show that the best accuracy is obtained when both the number of segments per word and the number of mixtures in each segment's GMM are kept as 4.

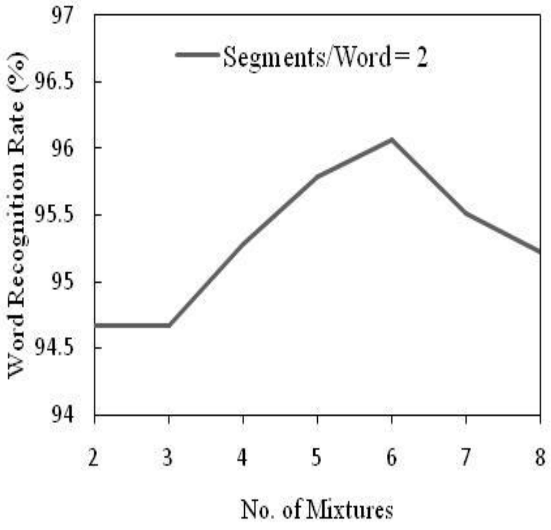

Figure 5.4: Word recognition rate for different number of mixtures when segments/word=2

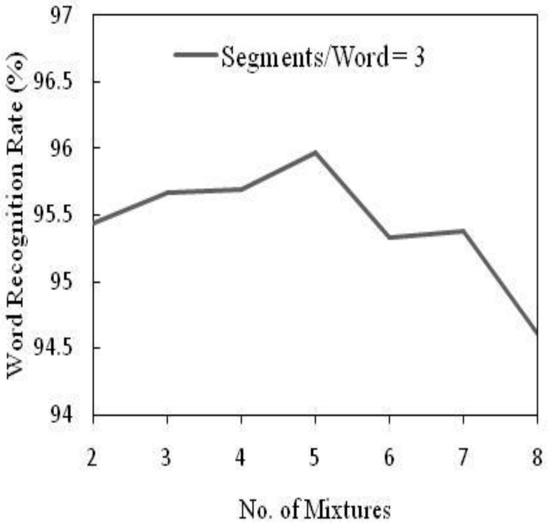

Figure 5.5: Word recognition rate for different number of mixtures when segments/word =3

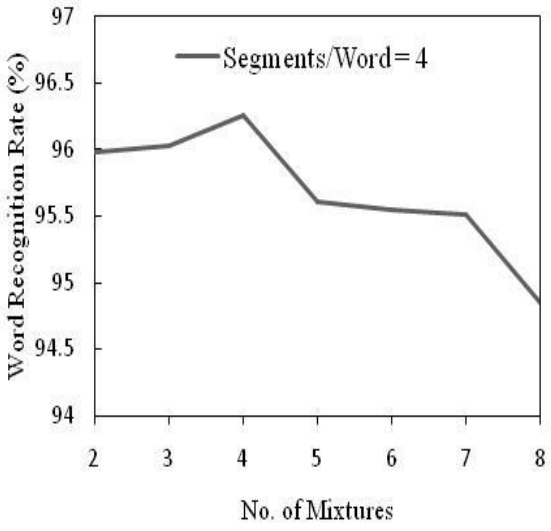

Figure 5.6: Word recognition rate for different number of mixtures when segments/word =4

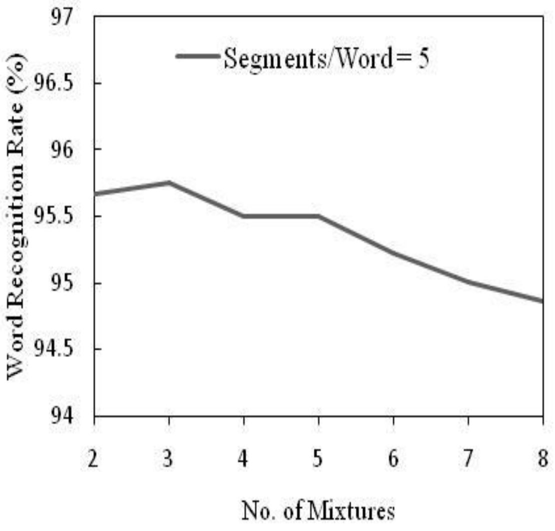

Figure 5.7: Word recognition rate for different number of mixtures when segments/word =5

Speaker Dependency: The results presented in Table 5.2 show a word recognition accuracy of 96.26%. It has to be noted that the same set of speakers were involved in the training and testing processes. It is known that GMM in general is interpreted as representing some speaker-dependent spectral shapes [25]. Hence it is assumed that all the speakers engaged in testing process will be having their speaker characteristics in the word model. It is worth investigating the behavior of our proposed technique for the scenario where speakers who are not part of the training set are subjected to the testing process.

No. of Speakers used in Training	No. of Utterances of each Word per Speaker Taken for Training	Approximate Duration of Training Speech for each Word Model	Word Recognition Rate (%) for Speakers from the Training Set	Word Recognition Rate (%) for Speakers outside the Training Set
12	1	6 seconds	95.32	93.03
8	1	4 seconds	91.44	83.51

Table 5.3: Word recognition with different sets of speakers for training and testing

The Table 5.3 presents the results obtained when the above said experiments are carried out. We followed the simple equal segmentation method and the number of segments per word and the number of mixtures are kept at 4 based on previous experiment results. Two different training sets are created - one involving 12 speakers and another with 8 speakers. Both the training sets are gender balanced. When the first set is used, word utterances from the remaining 4 speakers from the database are used for testing and when the second set is used, word utterances from the other 8 speakers are available for testing.

Upon first look, the results appear to show that our proposed technique does retain some form of speaker dependency in it. This can be seen from the decrease in accuracy when speakers outside training set are used in testing compared to the case where the same speakers from the training set are used. However, it can be observed in Table 5.3

that the loss of recognition accuracy is less when the amount of training speech is increased. When speakers outside the training set are used in testing, there is a drop of only 2.3% in accuracy in the first case where approximately 6 seconds of speech is used in training but the drop in accuracy is almost 8% in the second case with about 4 seconds of training speech. From the above observations, we can infer that for our proposed technique, a word model prepared using utterances of that word from a higher number of speakers will be more robust against test utterances from speakers outside the training population.

5.5 Multiple Models with Different Number of Sub Words

In the previous sections, we explained the sub-word constrained GMM technique for word recognition and presented the results. In the experiments carried out earlier, the number of segments per word in the model was kept the same for all the words in the set and the optimal number of segments was found to be 4. When words with very short duration are modeled with large number of segments per word, the speech feature vector distribution in each GMM in that word model will not be sufficient to provide information about the acoustic content. Also, in such cases, it is highly probable that few continuous GMMs in that word model represent features with similar acoustic characteristics. This adds more weight to the match score of words which are similar to that of the modeled word and could lead to incorrect identification. Examples for this case in the TI46 database vocabulary are the words GO and NO, which have only 2 phonemes in it. If these words are modeled with 4 segments per word, then the last 3 GMMs in the word model may correspond to the last phoneme, which is the same for both words. Hence for shorter words, modeling with fewer segments per word would be more appropriate to make a clear distinction between similar words. This scenario points to the difficult problem of finding the optimal number of segments for each word in the vocabulary. However, in this section, we propose a concept of having multiple models for each word, where each of the models is trained with different number of segments. Based on the scores obtained during the testing process, we subject the similar words into second round of testing with models trained using fewer segments per word, to make the decision of word identification. The categorization of similar words is done by setting a threshold in the score values. This process evaluates the above mentioned point that the incorrect identification that happens between similar short words, when they are modeled with a higher number of segments, can be overcome by modeling with fewer number of segments. In the following, we explain the steps involved in identifying a word using this technique and present the results.

5.5.1 Steps Involved in the Process

In the proposed technique, each word is represented with two time-ordered models. These two models are classified as main model and secondary model. The main model for each word is trained by segmenting the word into 4 sub word units. The number of segments is kept at 4 here because our earlier experiments reveal that it is the optimal one. The secondary model is trained by segmenting each word into 2 sub-word units.

The steps involved in the testing process are illustrated in Figure 5.8. The test utterance is first scored against the main model of all the words. It is assumed that in the case of unambiguous identification, the difference between the score obtained against the correct word model and the score obtained against rest of the models will be relatively high. In the figure, M_1 denotes the margin between the best score obtained S_{best} and the second best score S_{next}. If the margin $M_{1,}$ is higher than a predetermined threshold T_1, then the word against which the best score is obtained is identified as the uttered word. If not, then word that yielded S_{best} and the words whose scores differ from S_{best} by less than $T_{1,}$ are selected for scoring against the secondary models. M_2 denotes the margin between the best score S'_{best} and the second-best score S'_{next} obtained against the secondary models. If the second round of scoring improves the differentiation between the similar words, then the margin M_2 should be higher by a certain threshold T_2 than the margin M_1 obtained in the first round of scoring. If this condition is met, then the word that yielded S'_{best} is identified as the uttered word. If not, the word that yielded the best score in the first round of scoring against the main models is identified as the uttered word.

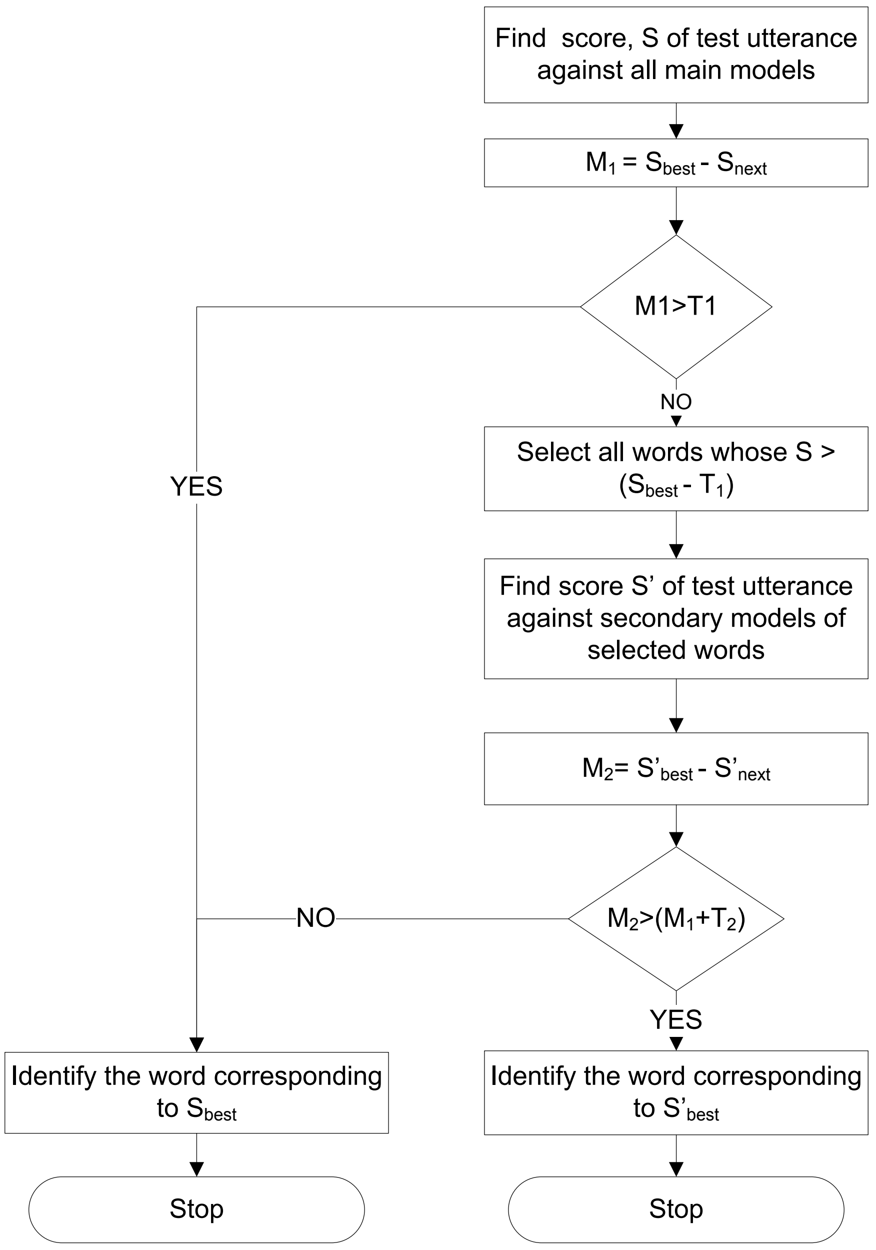
Find score, S of test utterance against all main models
$M_1 = S_{best} - S_{next}$
M1>T1
NO
YES
Select all words whose S > ($S_{best} - T_1$)
Find score S' of test utterance against secondary models of selected words
$M_2 = S'_{best} - S'_{next}$
$M_2 > (M_1 + T_2)$
NO
YES
Identify the word corresponding to S_{best}
Identify the word corresponding to S'_{best}
Stop
Stop

Figure 5.8: Testing process when word is represented with main and secondary models

5.5.2 Experimental Results

For training the two set of models, the same procedure used in the earlier experiments in this chapter is followed. 4 Mixtures are used to model each segment in both the main and the secondary models. The testing process is done according to the steps described in the previous sub-section. The values of T1 and T2 are varied and the results obtained for different combination of the values are presented below in Table 5.4.

Score-Margin Threshold for First Round of Scoring T1	Score-Margin Threshold for Second Round of Scoring T2	Word Recognition Rate (%)
2	0.2	96.85
	0.6	96.69
	1	96.62
3	0.2	96.87
	0.6	96.71
	1	96.63
4	0.2	96.93
	0.6	96.73
	1	96.64
5	0.2	96.93
	0.6	96.73
	1	96.64

Table 5.4: Word recognition using multiple models per word

Results show that for all combinations of the values of T1 and T2, there is a marginal increase in the word recognition accuracy compared to the result obtained earlier. The highest recognition rate, 96.93% is obtained when T1 is equal to both 4 and 5 and T2 is 0.2. As mentioned earlier, the secondary model used in this experiment has 2 segments per word and uses a 4-mixture GMM to model each segment. It has to be noted, that when the same training set-up is used in Section 5.4, the results obtained, as indicated in Figure 5.4, was 94.66% and is less than the result obtained here. This shows that using a small number of segments is effective only for some words in the vocabulary. This points to a future research challenge of developing a method to determine the optimal number of segments for each word as a superior alternative to the multiple-models concept explained here.

5.6 Summary

In this chapter, we proposed a technique to incorporate the time sequence information to improve GMM based isolated word recognition. We took an approach different from the conventional GMM method of globally modeling the probability density function (PDF) of all the speech feature vectors in the training data. We constrained the GMM to operate on segments or sub-word units of the word and represented the word as a sequence of GMMs corresponding to its sub-word sequence. In order to find a technique to segment the words, we discussed about the existing segmentation techniques and concluded that segmenting a word into a fixed number of segments without the knowledge of any linguistic information is beneficial and well-suited for our approach taken for word recognition. We examined two different methods for word segmentation.

Experiments were carried out using the TI20 directory of the TI46 database. The results showed that our approach substantially outperforms the conventional GMM approach and achieves a word recognition accuracy of 96.26%. Two important points observed from the results helped to keep our intention of minimizing the overall computational complexity intact. Firstly, the simple equal segmentation technique consistently outperformed a compute-intensive criteria-based segmentation technique. This nullifies the complex computations that are normally needed for segmenting a word. The second point was that the word recognition accuracy decreased when the number of segments per word and the number of mixtures were increased and the optimal values for the above were found to be 4. This again eliminates the need for more computations involving more number of models and mixtures in both the training and testing process. We also evaluated the concept of having multiple models per word and observed an increase of 0.7% in word recognition accuracy. Though this concept increases the storage and computation requirements, the increase in recognition rate shows that smaller words in the vocabulary can be better trained with even less number of segments.

From a system architecture perspective, it can be visualized that to implement our proposed technique, a single GMM engine that performs both modeling and testing can be invoked multiple times depending upon the number of segments per word used. The sub-word constrained GMM technique also lends well for a parallel implementation with multiple GMM engines processing several segments in parallel. In our planned set-up where speaker and word recognition is performed in the same system, the GMM engine employed for word recognition can be utilized for speaker recognition as well. In the next chapter, we discuss the GMM-based integrated system for speaker and word recognition.

Chapter 6

Integrated Speaker and Word Recognition System

6.1 Overview

In the last two chapters, we discussed the GMM based approaches for speaker recognition and word recognition. For word recognition, a novel technique, sub-word constrained GMM, was proposed. For speaker recognition, text-dependent and text-constrained set-ups were observed to yield strong results. From these results, it is evident that an integrated speaker and word recognition system based on GMM technique is feasible. In this chapter, we discuss two variants of such an integrated system. The first one is based on text-constrained speaker recognition and the second one is based on text-dependent speaker recognition. The rest of the chapter is organized as follows. We present the structure of the two versions of the integrated system in Section 6.2. In Section 6.3, we present the overall recognition accuracy obtained for these two versions and discuss the advantages and disadvantages. In Section 6.4, we evaluate the integrated system operating on a smaller set of commands. Section 6.5 summarizes this chapter.

6.2 Integrated Speaker and Word Recognition System

In this section, we describe the integrated speaker and word recognition system. The objective of the system is to take a word utterance as input and identify the uttered word and the speaker who uttered it. Both the recognition processes involved in this system, namely, word recognition and speaker recognition are based on the GMM technique as explained in the last two chapters. This process of using only the GMM technique simplifies the architecture of the system. From the architecture perspective, this integrated system will contain the following basic blocks.

- Speech capture unit
- Feature extraction unit
- Training unit
- Recognition unit
- Model database

Both the training unit and the recognition unit mentioned above use the same GMM engine and it is worth noting that these two units can be used for both word and speaker modeling/recognition.

For performing word recognition in the system, we adopt the technique proposed earlier which incorporates the time sequence information in GMM modeling. For speaker recognition based on GMM, our earlier evaluations showed that text-dependent speaker recognition provided better results compared to text-constrained one. However, it is worth investigating the impact of both the speaker recognition methods in the integrated system. Hence we present two variants of system, one with text-constrained speaker recognition and the other with text-dependent speaker recognition.

6.2.1 Integrated System Based on Text-Constrained Speaker Recognition

The first version of the integrated system we describe here is based on text-constrained speaker recognition. The outline of the structure of this system is presented in Figure 6.1. Both the speaker recognition and word recognition processes involved in this system are independent of each other. The steps involved in identifying the uttered word and the speaker who uttered it are described as follows. The word uttered is captured through the speech capture unit and in the next step the feature vectors of the word are extracted. The feature vectors are used as input for the word recognition process, which segments the feature vector sequence and scores the vectors against the word models in the database to identify the word. Next, the feature vectors are used for speaker recognition process, where the same GMM scoring process as in word recognition is used, which identifies the speaker using the speaker models in the database.

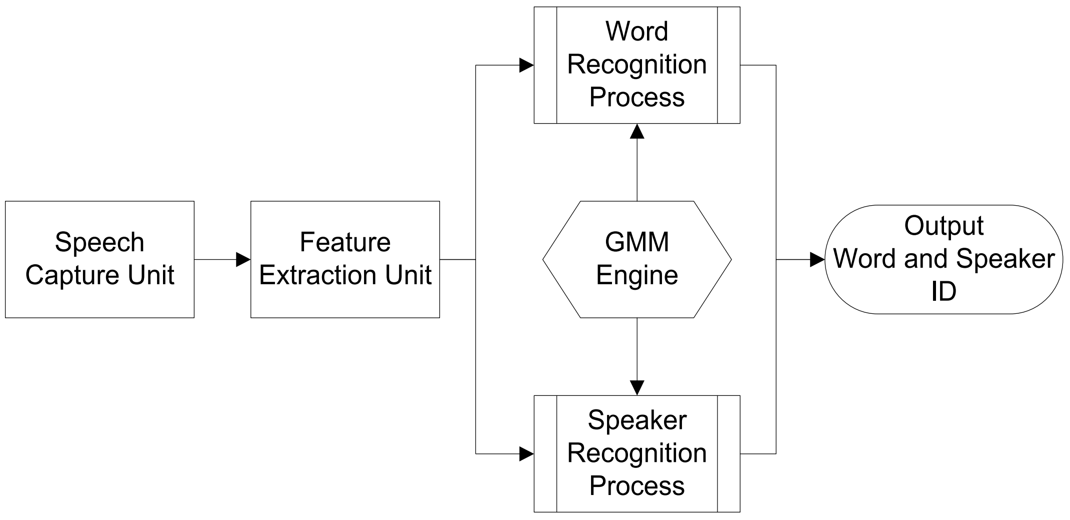

Figure 6.1: Integrated system based on text-constrained speaker recognition

The steps in the recognition process of the system described above, reveals the following advantage the system possesses. Since, both the word recognition and speaker recognition processes function separately using the same set of feature vectors, it is not necessary for one process to wait till the other is completed. Instead, both the processes can possibly be fine tuned to function in parallel. This increases the processing rate of the system.

6.2.2 Integrated System Based on Text-Dependent Speaker Recognition

The outline of the integrated system based on text-dependent speaker recognition is presented in Figure 6.2. Unlike the text-constrained version described earlier, this system follows a two-step approach. After extracting the feature vectors of the input word, the word recognition is carried out first. In the next stage, based on the word identified, the speaker models, trained using that particular word, are selected for the speaker recognition process. The speaker identification obtained in this stage and the word identified in the previous stage provides the final result of the system.

In this system, the total number of speaker models in the database will be more than the system based on text-constrained speaker recognition. For each speaker, separate speaker models should be trained using each word in the vocabulary separately. However, it has to be noted that the number of speaker models that will be used for determining a speaker's identity will be the same as the text-constraint version and it depends on the number of speakers enrolled in the system.

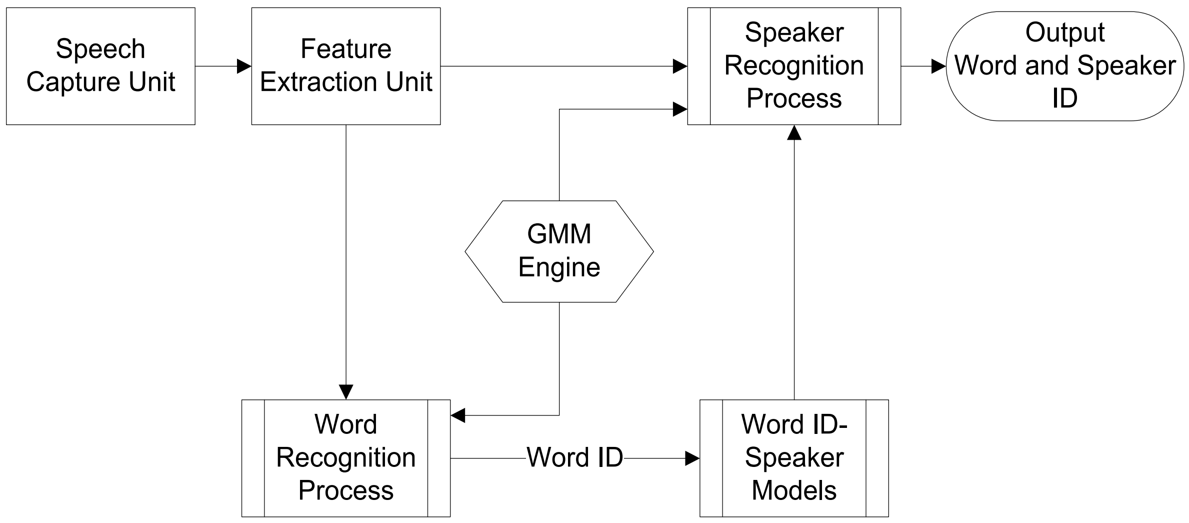

Figure 6.2: Integrated system based on text-dependent speaker recognition

6.3 Experimental Results

Experimental Set-up:

The TI20 directory of the TI46 database and the training procedure used in the last two chapters are used in this experiment as well. We evaluated GMM based speaker recognition and word recognition in the last two chapters using the same database and found out the optimal values for various modeling parameters. Hence, for the integrated systems to be evaluated here, we consider the same values for the modeling parameters. For the 20 word models, the number of segments per word and the number of mixtures in each sub-model are kept at 4. For text-constrained speaker recognition, the 16 speaker models are trained using 10 utterances of all the words with 32 mixtures per model. For text-dependent speaker recognition, each speaker model is trained as an 8-mixture model using 10 utterances of a word. Utterances from the TEST session are used to train the models and the 3186 utterances available in the TRAIN session are used for identification process.

6.3.1 Results and Discussion

Integrated System Type	Word Recognition Rate (%)	Speaker Recognition Rate (%)	Overall Recognition Rate (%)
System 1: With Text-Constrained Speaker Recognition	97.18	96.61	94.03
System 2: With Text-Dependent Speaker Recognition	97.18	98.74	96.55

Table 6.1: Overall recognition rate for the two versions of the integrated system

The results obtained from the experiments are listed in Table 6.1. The overall recognition rate shown in it denotes correct speaker and word identification for a test utterance. It has to be noted that the change in word recognition accuracy from the results obtained in the previous chapter is due to the fact that we interchanged the training and testing sessions of the speech database for this experiment. The results presented in Table 6.1 shows that the integrated system based on text-dependent speaker recognition results in better overall recognition accuracy. Clearly, the increase in speaker recognition accuracy because of the two-step approach and the use of word-specific speaker models in that system has contributed to the better results.

The two-step approach causes the system to depend more on the word recognition accuracy for the overall performance. As seen in Chapter 4, the text-dependent speaker recognition has a recognition rate of 99.28% and therefore it provides very reliable speaker recognition in the integrated system if the word is recognized correctly. This can be observed from Table 6.2, where a high speaker recognition rate of 99.35% has been achieved for cases where the word is identified correctly. On the other hand, when the word is wrongly identified, it causes the speaker models corresponding to the wrong word to be used in the speaker recognition process resulting in a markedly inferior speaker recognition accuracy of 77.78%.

Word Recognition Outcome	No. of Instances (Word Recognition)	No. of Instances where Speaker Correctly Identified	Correct Speaker Identification (%)
Word recognized correctly	3096	3076	99.35
Word recognized wrongly	90	70	77.78
All	3186	3146	98.74

Table 6.2: Effect of word recognition accuracy on speaker identification in the text-dependent version of the integrated system

Though the text-dependent version of the integrated system proved to be better in terms of recognition accuracy, it has the drawback of higher memory requirement compared to the version that incorporates text-constrained speaker recognition. This is because of the need to store a higher number of speaker models depending on the size of the vocabulary. However, this is not a major setback since the potential practical applications of the integrated system typically deal with a small set of commands. As stated earlier, a fine tuning of the scoring process in the text-constrained version can enable both the word recognition and speaker recognition processes to operate in parallel resulting in higher processing speed. While this is not possible in the system with text-dependent speaker recognition, it is worth noting that the number of mixtures used in the text-dependent speaker models will be typically small compared to the text-constrained speaker models. In the experiments described above the number of mixtures in the speaker models for the text-constrained and text-dependent versions are 32 and 8 respectively. Therefore, the scoring in the speaker identification process in the latter is almost 4 times faster than the former. Because of the above mentioned points, we chose the integrated system with text-dependent speaker recognition for the subsequent work in this project.

6.4 Integrated System Operating on a Small Set of Commands

The development of the integrated speaker and word recognition system explained above is done in line with our main objective of making it suitable for resource-constrained embedded platforms like mobile devices. We evaluated the system with a vocabulary that contains 20 words. However, in typical mobile and embedded applications, only a smaller set of words, usually consisting of a few commands, is used. Therefore, before incorporating the system into a mobile application, it is beneficial to evaluate its performance for a smaller set of commands. In this section, we evaluate the integrated system's performance using the following three different set of words, chosen from the TI20 vocabulary.

Set 1
{ZERO, ONE, TWO, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT, NINE}

Set 2
{ENTER, ERASE, HELP, NO, RUBOUT, YES}

Set 3
{START, GO, STOP, REPEAT}

The sets are formed with different sizes so that the change in performance of the system for different size of vocabulary can be analyzed. Set 1 contains the 10 digits which are normally used in voice dialing applications. The six words in Set 2 are selected so that there are no similar words in it. Whereas Set 3 is formed by including two commonly used commands, START and STOP, which contain acoustically similar parts.

6.4.1 Experimental Results

Experimental Set-up:

The same training set-up used in the experiment mentioned in the last section is used here. The integrated system with text-dependent speaker recognition is used in this experiment. All the 16 speakers in the database are included in the experiment for speaker recognition. While evaluating each of the three sets mentioned earlier, the test utterance will be one of the words from the set evaluated. Altogether for Set 1, there are 1594 test utterances. For Set 2, there are 955 test utterances and for Set 3, the number of test utterances is 637.

Experimental Results:

Command Set	Word Recognition Rate (%)	Speaker Recognition Rate (%)	Overall Accuracy (%)
Set 1	98.93	98.99	98.30
Set 2	99.69	99.58	99.37
Set 3	98.43	98.59	97.17

Table 6.3: Integrated system operating on a small set of commands

The results from the experiments are presented in Table 6.3. From the table it can be observed that for all the three sets, the overall recognition accuracy has bettered the 96.55% accuracy obtained in the last experiment when 20 words set is used. Similarly, the word recognition rate for all the three sets has also increased. When the number of words in the set is reduced from 10 in Set 1 to 6 in Set 2, the recognition rate increases from 98.3% to 99.37%. Interestingly, Set 3, which is the smallest of the sets and contains only 4 words, has resulted in lower performance compared to the other two sets. However, as mentioned earlier, two of the words in the set have similar acoustic classes and hence the possibility of incorrect recognition is more. Apart from this, the results clearly favor the implementation of the integrated speaker and word recognition system for applications in mobile devices.

6.5 Summary

This chapter introduced the GMM based integrated speaker and word recognition system. In the system, for word recognition, we used the technique of time-ordered GMM word models explained in Chapter 5. For speaker recognition, as seen in Chapter 4, two different schemes were found to provide good results. Hence two versions of the integrated system were proposed based on text-constrained speaker recognition and text-dependent speaker recognition respectively. We analyzed the two systems and established that the text-dependent version provides better performance. The knowledge of the word identified was utilized to improve the speaker recognition accuracy by the two-step approach followed in the system based on text-dependent speaker recognition. We also identified that while the storage requirement of the text-dependent version will be relatively higher, the scoring in the speaker recognition process will be faster because of the smaller number of mixtures it uses in the speaker model. Because of these reasons, the integrated system with text-dependent speaker recognition is chosen for the subsequent work. Experiments carried out to evaluate the performance of this integrated system for a smaller set of words provided results better than those obtained for a larger set of words thereby proving its suitability for typical mobile applications. In the next chapter, we implement this integrated system as a mobile application and test it in real-life environments.

Chapter 7

Mobile Application for Speaker-Aware Isolated Word Recognition

7.1 Overview

In the last chapter, we discussed the GMM based integrated speaker-and-word recognition system and established that the two-step system with word recognition based on the technique proposed in Chapter 5 and speaker recognition based on the text-dependent approach yields better performance. The optimal values for various processing parameters are also established by the experiments carried out in the previous chapters. In this chapter, we develop the above system as an application suitable for mobile devices. The rest of the chapter is organized as follows. Section 7.2 gives the description of the platform chosen. Section 7.3 explains the development of various modules in the mobile application. In Section 7.4, the implementation results are presented. We summarize this chapter in Section 7.5.

7.2 Platform Chosen

The integrated speaker-and-word recognition system is developed as an Android application. The choice of the Android platform is motivated by the huge growth in the usage of smartphones with Android support in recent times. The market share of Android in smartphone sales in the year 2010 is 22.7% which is an enormous 888% increase compared to the previous year [92]. In the first quarter of the year 2011, Android has become the market-leading smartphone platform with 36% market share [93].

Android is an open source software stack built on top of a Linux kernel for mobile devices that includes an operating system, middle-ware and key applications. The Android Software Development Kit (SDK) provides the tools and the Applications Programming Interfaces (APIs) necessary to develop applications on the Android platform using the Java programming language. Any mobile device that supports Android can implement and run an Android application. In the following, we provide an overview of the Android architecture and the development environment.

Platform / Company	Market Share (%)
Android	36.0
Symbian	27.4
Apple iOS	16.8
Research in Motion (RIM)	12.9
Microsoft	3.6
Other OS	3.3

Table 7.1: Worldwide smartphone sales to end users in the first quarter of 2011

7.2.1 Android Architecture

The Android software stack is a multilayered environment, where each layer groups together several programs that support specific operating system functions [94]. Figure 7.1 shows the layers in the Android environment and the major components in each of them.

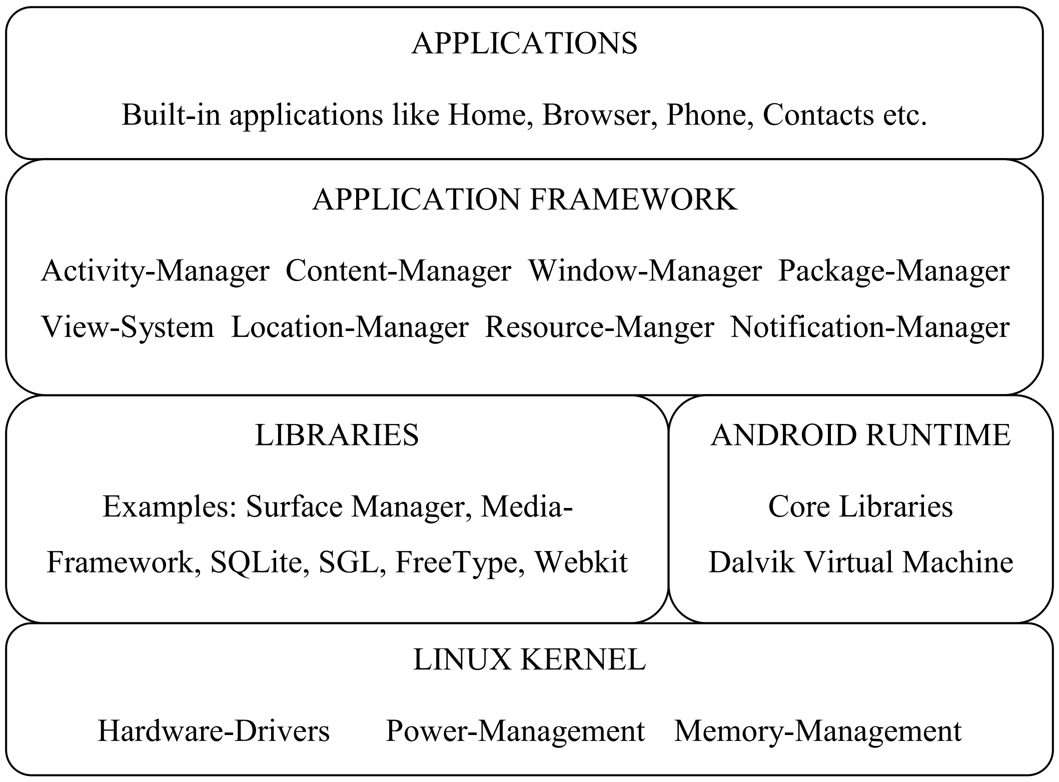
APPLICATIONS
Built-in applications like Home, Browser, Phone, Contacts etc.
APPLICATION FRAMEWORK
Activity-Manager Content-Manager Window-Manager Package-Manager
View-System Location-Manager Resource-Manger Notification-Manager
LIBRARIES
Examples: Surface Manager, Media-Framework, SQLite, SGL, FreeType, Webkit
ANDROID RUNTIME
Core Libraries
Dalvik Virtual Machine
LINUX KERNEL
Hardware-Drivers
Power-Management
Memory-Management

Figure 7.1: Android architecture

Applications Layer

The top layer of the Android environment consists of the core Java applications such as dialer, address book, browser etc., which can be accessed through the user interface provided. Mobile device users will normally deal only with this layer but application developers or hardware manufacturers should have the knowledge about the remaining three lower-level layers.

Application Framework

This layer provides the functions that manage the basic APIs of a mobile application like resource management, content management, user interface management etc. The developers have full access to all the APIs even if it is used by the core applications in the top layer. This application framework layer is designed to simplify the reuse of the components and therefore any application can publish its own framework APIs and any other application may then make use of it. This layer contains important functions including

Views:	A rich and extensible set that can be used to build the user interface of an application.
Content Providers:	This enables applications to access data from other applications such as Contacts in mobile phones.
Resource Manager:	This provides access to non-code resources such as graphics and layout files.
Notification Manger:	This enables all applications to deal with custom alerts.
Activity Manager:	This manages the lifecycle of the applications.

Libraries

The next layer of the Android architecture includes a set of C/C++ libraries used by various components of a device that supports Android. For example, the media framework library supports playback and recording of various audio, video and picture formats. Other important libraries in this layer includes the System C library, which is an implementation of the standard C system library tuned for embedded Linux-based devices, Surface-Manager, which manages access to the display subsystem and composites 2D and 3D graphic layers from multiple applications, and SQLite, which is a powerful and lightweight relational database engine available to all applications. All the libraries in this layer are exposed to developers through the application framework.

Android Runtime

The Android Runtime layer is located on the same level as the Libraries and it contains a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language. It also includes the Dalvik Virtual Machine (DVM). Every Android application runs in its own process, with its own instance of the DVM. It has to be noted that Android applications are programmed using the Java programming language. A tool called 'dx' is used to convert the Java class files into '.dex' format, which is optimized for memory footprint. The DVM executes the files that are in the '.dex' format. The DVM relies on the Linux kernel for underlying functionality such as threading and low-level memory management.

Linux Kernel

The base of the Android software stack is the kernel. The Linux version 2.6 is used to build this kernel. The kernel acts an abstraction layer between the hardware and the rest of the software stack. Also, Android relies on the kernel for core system services such as security, memory management, process management and network stack. The drivers for the various hardware components in the device are also managed by this kernel.

7.2.2 Android Development Environment

Android applications, like most mobile phone applications, are developed in a host-target development environment. It is developed on a computer with abundant resources and then transferred to the mobile device. To develop android applications, the main components needed are the Java Development Kit (JDK), Android SDK, Eclipse Integrated Development Environment (IDE) and the Eclipse Android Developer Tools (ADT) plug-in. The JDK provides a collection of Java programming tools and the Android SDK enables developers to create applications for the Android platform by providing the development tools, emulator, and required libraries. Eclipse is a software development environment for building, deploying and managing software across the entire software lifecycle. It provides a core of services for controlling a set of tools working together to support programming tasks. Though there are a few other IDEs that can support Android application development, Eclipse is recommended since Android has the official plug-in support for it.

7.3 Integrated Speaker-and-Word Recognition System as an Android Application

In this section, we present the integrated speaker-and-word recognition system developed as an Android application. The salient features of Java programming involved in the development and the Android libraries and frameworks used in various modules of the application are described. The entire application of the integrated speaker-and-word recognition system is divided into 4 modules. Figure 7.2 shown below illustrates the modules and the execution flow.

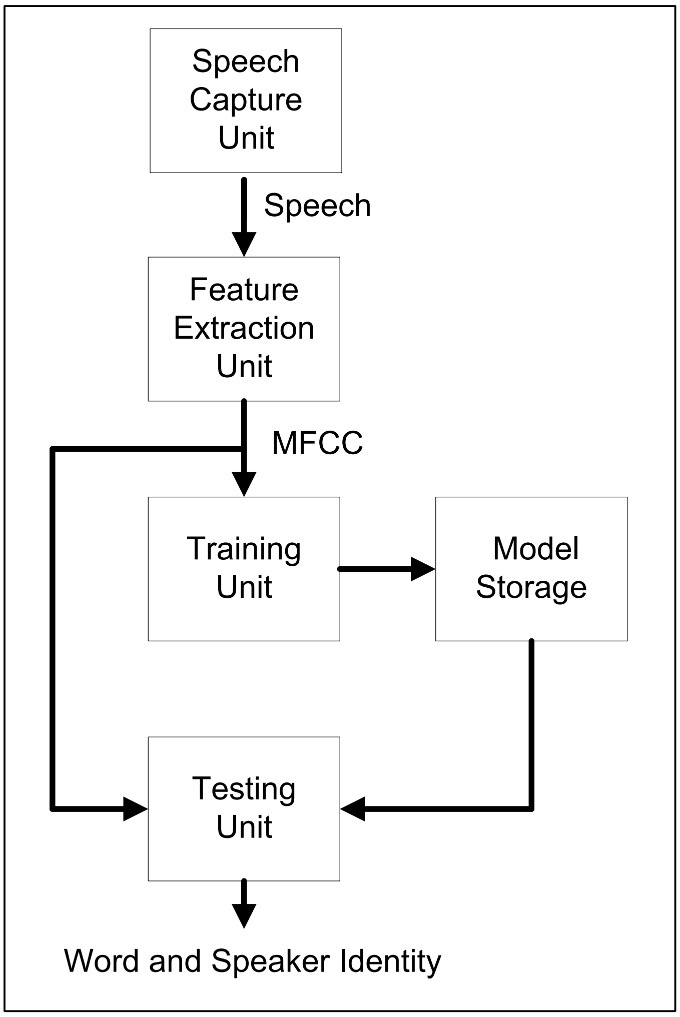

Figure 7.2: Overview of the Android application for speaker-and-word recognition

The Android application is developed using the Java programming language. Java is a general-purpose, concurrent, class-based, object-oriented language that is specifically designed to have as few implementation dependencies as possible. In line with the concept of Java, our application is also programmed in an object-oriented manner. A collection of objects and classes for various functions in the application is developed and all the 4 basic modules of the application use the necessary objects from this collection. In Android application development, some of the Java libraries, especially the ones associated with the internal hardware of mobile devices like the microphone, are overwritten by the Android specific APIs. In our work, such a scenario occurs in the speech capture unit, where word utterances by a speaker have to be recorded for either training or testing. The Java Sound package normally used to record speech in Java programming is replaced with the Android APIs as explained in the following.

7.3.1 Speech Capture Unit: Android APIs for Speech Recording

At present, there are two different APIs that can support audio recording in Android. They are MediaRecorder [95] and AudioRecord [96].

MediaRecorder: The MediaRecorder API can be used to record audio from the microphone of the mobile device to a file on the memory card. The recorded audio would be in MPEG4, RAW, AMR or 3GP format. Though it is easy and straight forward to use, it also has disadvantages with respect to our requirement. The various recording parameters cannot be modified and the audio buffers cannot be accessed during recording. Also, the audio is recorded in a compressed format and hence for speech processing an additional step of uncompressing the data has to be carried out. This increases the time taken between the speech capture and the speaker-and-word identity output in our application.

AudioRecord: The AudioRecord API for audio recording is used in our application development as it overcomes the shortcomings of the MediaRecorder API mentioned above. The recording is done in uncompressed WAV format and the audio buffers can be accessed during the recording. Also, all the recording parameters like the sampling rate, sample size, channel type and size of the buffer can be set to the desired values.

7.3.2 Model Storage

The unit Model Storage shown in Figure 7.2 indicates the area where the speaker and word models are stored after training. These models have to be accessed by the testing unit to perform the identification of the word and the speaker. Therefore, the storage location and the organization of the models should be made available to the application. Android provides a relational database called SQLite and this can be utilized to store data related with the applications. The database created and used by the application is coupled along with it and hence when an application is ported to a device, the data in the database are also transferred to the device.

In our work, instead of using the SQLite database, we followed a simple way of storing the models in the memory card of the device and reading them from it for scoring in the testing unit. Java data output and input streams are used for this purpose. A data output stream lets an application write primitive Java data types to an output stream in a portable way. A data input stream lets an application read those data types from an underlying input stream in a machine-independent way. The procedure followed for model storage makes the stored model free from any control by the application and hence when the application is ported to a mobile device, the models should also be ported to the device separately. Though this is not a standard and data-security-compliant procedure for maintaining data storage, we followed this convenient approach since our main focus was only on validating the performance of the integrated speaker-and-word recognition system as a mobile application.

7.3.3 Computation Speed-up through Inter-Module Parallelism

Figure 7.2 shows the input for the three units, feature extraction, training and testing. While the training and testing units can start their function only after receiving all the MFCC vectors, the feature extraction unit need not wait till the entire word utterance is captured. Since we use the AudioRecord API for recording speech in the speech capture unit, it is possible to access the samples while capturing the speech. It is known that the feature extraction is done for each frame of very short duration. In our work, we used 20 msec speech frames progressing at a rate of 10 msec. During speech capture, when the first frame, that is 20 msec of speech, is captured, the feature extraction unit should be notified to process that frame and extract MFCC for it. This notification is repeated for every speech frame that is being captured. During the feature extraction process, the speech capture unit should not be suspended and should be allowed to capture speech to prevent loss of speech samples. This parallel functioning is achieved in our work using the multithreading concept of Java, which enables several parts of a program to run in parallel. Java uses threads, which are lightweight processes part of one program that can access shared data, to facilitate parallel operation. In our application, the speech capture unit and the feature extraction unit are designed as two separate threads and the buffer where the samples are stored during speech capturing is shared between the two threads.

7.3.4 User Interface Management of the Application

Android provides an application framework API known as Views, which contains classes that handle screen layout and interaction with the user. In our work, the various processing parameters in all the units are fixed according to the findings from the experiments carried out in the previous chapters. Therefore the main interactions between the user interface screen and the application are the activation of the application by the user, the notification to the user for speaking and the display of speaker and word identification results in the screen. We followed the linear layout class provided by the View API, which aligns all the components like text fields, buttons etc. in a single direction - vertically or horizontally, depending on how the orientation attribute is defined.

7.3.5 Device Support

The Android application is deployed to a device along with an xml file named AndroidManifest. This file contains the necessary configuration information to properly install the application on the device. It includes the required class names and types of events the application is able to process, and the required permissions the application needs to run. In our application, the permissions to use the microphone for recording speech and to write and read files from the memory card have to be specified.

Android applications run only on devices that have support for Android operating system. The speaker-and-word identification application involves many floating point operations and hence the device should have a processor with floating point unit to support the successful running of the application. The device selected for testing the implementation of the algorithm is presented in the following section.

7.4 Implementation Results

We selected the Motorola CHARM mobile phone that runs the Android operating system to test the implementation of our algorithm. The important specifications of the mobile phone are given below.

Operating System:	Android OS, version 2.1
CPU:	ARM Cortex-A8 600 MHz processor
Memory:	512MB RAM

The implementation is analyzed for the accuracy of speaker-and-word identification and the computation time taken by the various modules. The findings are presented in the following.

7.4.1 Performance Analysis

In order to evaluate the performance of the Android application for speaker-and-word recognition in real-life environments, an evaluation is carried out using five speakers and a set of five words. All the five speakers are male, non-native speakers of English, in the age group of 28 to 35. The set of command words used are *ENTER*, *START*, *GO*, *REPEAT* and *ERASE*. The recording of speech for training is done in a closed room with a quiet background environment. Testing is carried out a week after the training session and is done partly in the same room and partly in a typical office environment with minimal background noise.

For both training and testing, the speech is captured using the internal microphone of the mobile device with mono channel recording and the speech is sampled at the rate of 8 KHz with 16 bits per sample. The best training set-up found from the results in Chapter 4 and Chapter 5 is followed here. For speaker model training, each speaker-word model is trained as an 8-mixture GMM by using 10 utterances of that word. For word model training, each word is segmented into 4 segments and each sub-model is modeled as a 4-

mixture GMM. 3 utterances of each word uttered by each speaker are used for training the word models. The training in general is done by using 10 iterations of the k-means algorithm to initialize the model and using the BA scheme explained in Chapter 3 to estimate the final model. For testing, each speaker uttered each of the words 10 times resulting in 250 test cases for speaker-and-word identification. The feature extraction set-up for training and testing is the same. 20 MFCC feature vectors are calculated from a 20 msec speech window progressing at a rate of 10 msec. The results are presented in Table 7.2.

Total Test Cases	Word Correctly Identified	Word Recognition Rate (%)	Speaker Correctly Identified	Speaker Recognition Rate (%)	Speaker and Word Correctly Identified	Speaker-and-Word Recognition Rate (%)
250	245	98.0	231	92.4	229	91.6

Table 7.2: Recognition accuracy of the Android application for speaker-and-word recognition

An overall recognition accuracy of 91.6% is obtained from the experiments. The speaker recognition rate is 5.6% less compared to the word recognition rate. Analyzing the performance of the individual words (Figure 7.3) show that only for two words in the list, ERASE and START, the speaker recognition rate is considerably lesser. For ERASE, the word recognition rate is 45/50 and since the speaker recognition depends on the indentified word, the fall in accuracy (41/50) is not surprising. For the word START, the speaker recognition rate is 44/50 even though 100% word recognition accuracy is obtained. It has to be noted that this word also produced the lowest text-dependent speaker identification performance (96.86%) among the 20 TI46 database words used in the experiments in Chapter 4 (Table 4.4). It is worth noting that both the words yielding lower speaker recognition accuracy, ERASE and START, contain the 's' sound, which is an unvoiced fricative produced without the vibration of vocal chords. Unvoiced sounds are known to exhibit noise-like randomness and are inadequate for distinguishing speakers. It can be observed from Figure 7.3 that 4 words produced 100% word recognition accuracy and the word GO, yielded correct speaker-and-word recognition all the times.

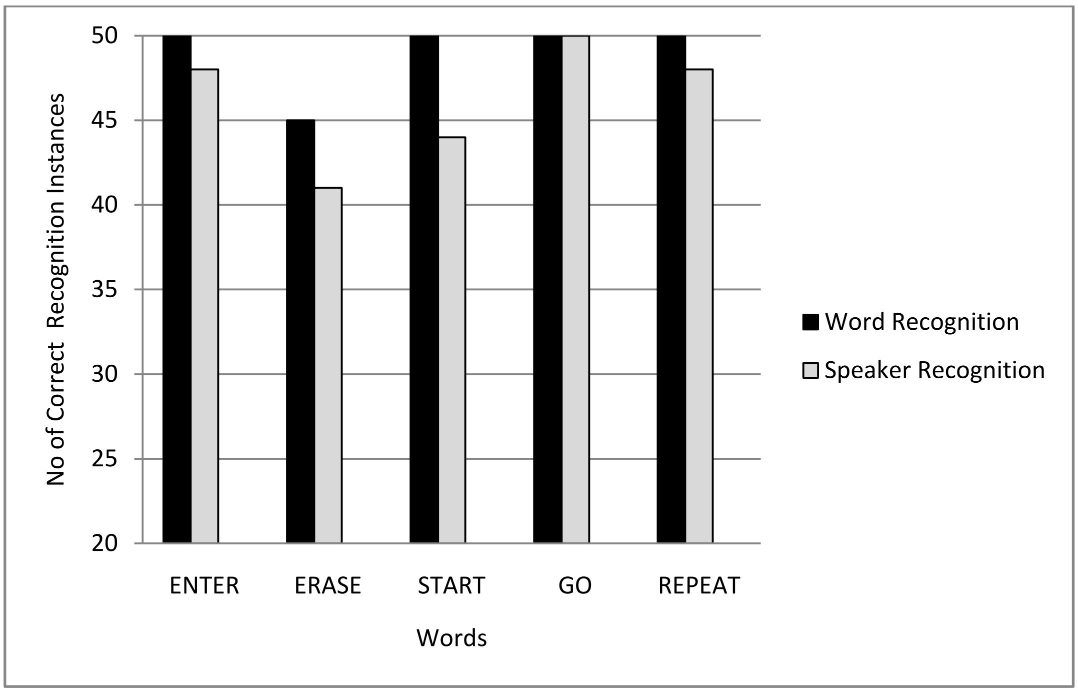

Figure 7.3: Speaker and word recognition performance for each word utterance

7.4.2 Execution Time Analysis

Process	Average Execution Time (msec)
FFT and Mel-Filter bank (per frame)	10.68
Log and DCT (per frame)	0.47

Table 7.3 Average execution time for the modules in MFCC calculation

The execution time of the steps in the major modules is summarized in the following tables. Table 7.3 shows the time taken for MFCC calculation for a single frame. The total calculation time is 11.15 msec. The overall time required to calculate the MFCC for an entire word depends on the number of frames in that word.

Table 7.4 presents the execution time for the initial model estimation, final model estimation and the scoring processes for an input utterance with 48 speech feature vectors. The number of mixtures per GMM is kept as 4 and 8. For the given input utterance, the total training time is 179 msec for estimating the 4-mixture GMM and 306 msec for the 8-mixture GMM. However, it has to be noted that multiple utterances of a word are considered for training speaker and word models and hence the overall training time depends on the total number of speech feature vectors processed.

No of Speech Vectors Processed	Process	Average Execution Time (msec)	
		Mixtures/ Model =4	Mixtures/Model =8
48	Initial Model Estimation (10 iterations of k-means)	142.90	236.10
	Final Model Estimation	35.84	69.91
	Scoring	16.80	32.64

Table 7.4: Average execution time for the modules in training/testing a model

A typical word utterance will be around 0.5 seconds in duration and according to our set-up of extracting MFCC from 20 msec speech window progressing at a rate of 10 msec, each utterance is also expected to have around 50 speech vectors. Hence the scoring time shown in Table 7.4 is typical for obtaining the match score against a model for an utterance. The total time taken to identify a word or speaker depends on the total number of models considered for scoring.

In the following, we provide an indication of the time required by the mobile implementation to perform speaker-and-word identification for a typical word utterance. The 48 speech vectors shown in the above table is obtained from an utterance of the word START in our evaluation with a set of 5 speakers and words. We used 4 segments per word in word modeling and hence there are 20 sub-word models with 4 mixtures each. Since the test utterance is segmented into 4, the number of speech vectors to be processed for scoring is reduced by a fraction of 4. Hence, it will approximately take one fourth of the above reported 16.8 msec time for scoring against a sub-word model and the time for obtaining the total match score against a whole word will be equal to 16.8 msec. Therefore, the time taken for word identification after scoring against all the models will be 84 msec (16.8 × 5). For speaker models, we used 8 mixtures and hence the time taken for speaker identification will be 163.2 msec (32.64 × 5). In total, after the speech is captured, the time taken for speaker-and-word identification will be 247.2 msec in addition to the time required to fetch the models from the database.

7.5 Summary

In this chapter, we discussed the development of the integrated speaker-and-word recognition system as an Android application. The multilayered architecture of the Android operating system was explained in detail. The Android application was developed using the Java programming language in an object oriented manner. The application was divided into four major modules, namely, speech capture unit, feature extraction unit, training unit and testing unit. To speed-up the output of the recognition process, the speech capture unit and the feature extraction unit were made to function in parallel using the multithreading concept of Java. The Android-specific AudioRecord API, that allows the captured speech samples to be accessed for parallel processing, was used for capturing the speech.

The application was deployed in a mobile device with ARM Cortex-A8 600 MHz processor and 512 MB RAM for testing the implementation. 5 speakers and a set of 5 command words were used for the evaluation and an overall speaker-and-word recognition rate of 91.6% was obtained. The typical execution times of the various modules in the application were also presented in this chapter.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

In this research work, an integrated speaker and word recognition system to recognize a spoken word and the speaker who uttered it has been developed. A study of the existing systems of similar type revealed that the word recognition and speaker recognition processes are carried out through different techniques that best match the processes. This inhibits the implementation of the system in devices with limited resources because of the high computational complexity involved. A reduction in the complexity of the above said systems can be realized if they are made to utilize a single technique for both speaker and word recognition. The literature survey carried out revealed that among the robust stochastic modeling techniques, GMM has fewer computations and has the potential to suit both speaker and isolated word recognition. Hence GMM was selected as the base of our algorithm and a rapidly configurable evaluation platform to expedite the GMM based experiments was developed. We envisage that this platform will also serve as a useful tool for future researches undertaken in this field.

The challenge in using GMM for speaker recognition is that the recognition has to be carried out using a word utterance of very short duration. One approach to overcome this problem is to constrain the training process to only the words in the vocabulary. Our experimentation of this text-constrained approach reveals that increasing the number of utterances of each of the words in the training set provides high robustness. The performance of this approach depends on the number of words in the vocabulary. If more number of words is used, the recognition process has to cover a large acoustic group and hence the accuracy of recognition will get reduced. We proposed and evaluated an improvement to text-constrained speaker recognition by grouping the similar sub-word units that constitute the words in the vocabulary. Although in the current form, the technique yielded only a performance similar to that of text-constrained speaker recognition, we believe that the recognition rate can be potentially improved by exploring alternative schemes for segmenting words into sub-words and for grouping them. We have identified this as a key area for future work spurred by this research. Another approach to solve the speaker recognition problem using a single word utterance is to carry out the recognition in text-dependent mode. Though this approach requires the speaker to be trained individually for each of the words in the vocabulary, our results show that it provides robust speaker recognition with 99.28% accuracy.

Extending GMM to word recognition posed a significant challenge because the time sequence information, which is vital for recognizing a word, is lacked by GMM. Therefore, a new technique to incorporate temporal information of speech in GMM is proposed. Each word is represented as a time-ordered sequence of GMMs, where the GMM sequence corresponds to the sequence of sub-word units in the word. GMM-based recognition is confined to these sub-word units by segmenting the test utterance into a definite number of segments. This method of incorporating temporal information is simple and does not require any linguistic/phonetic information to delineate the sub-word units or information on inter-sub-word transition probabilities. This technique improved the word recognition accuracy from 90.6%, obtained by the conventional GMM approach, to 96.26% for a 20-word vocabulary and further increased it to 96.93% when multiple time-ordered models per word are used. It is worth noting that the above increase in the recognition accuracy has been achieved without any significant increase in computational complexity. While typical segmentation algorithms used to segment the words are computationally complex, we have utilized a simple, yet effective, segmentation method that operates at the acoustic level and segments the words into a predetermined number of equal parts.

With GMM being successfully applied to word recognition, it is evident that an integrated system of speaker and word recognition can be designed using the GMM technique. After evaluating the integrated speaker-and-word recognition system with text-constrained and text-dependent speaker recognition approaches, the text-dependent speaker recognition approach is chosen in order to realize a robust system. Such a system recognizes the uttered word first and then identifies the speaker based on the speaker models prepared using the recognized word. This approach makes the speaker recognition process rely on the correctness of word recognition. Though the memory required for the speaker models in this system is relatively high because of the text-dependent speaker recognition approach, practical speaker-and-word recognition applications typically use only a small set of words and hence this drawback is unlikely to be a major concern. Having developed the integrated speaker-aware word recognition system, we ported the algorithm as an application into an Android-based mobile device operating with an ARM Cortex-A8 600 MHz processor. Experiments in real-life situations have been successfully carried out with a set of 5 words and 5 speakers and an overall recognition accuracy of 91.6% has been obtained. While the suitability of the proposed technique for resource-constrained mobile/embedded devices has been demonstrated, we believe that the recognition rate could be potentially improved through further research on this topic as outlined in the following section.

8.2 Future Work

In the following, we present a few suggestions for further developing the ideas propounded in this thesis. It is envisaged that these suggestions will propel further research and development activities in this domain.

Improving the robustness of text-constrained speaker recognition: In our evaluations with the TI46 corpus, text-dependent speaker recognition yielded a recognition accuracy of 99.28% compared to 96.6% for text-constrained speaker recognition. On the basis of this result, the text-dependent approach was adopted for the integrated system. However, text-constrained speaker recognition offers a few advantages such as making the word recognition and speaker recognition processes independent of each other. Also, it eliminates the need for storing multiple word-specific speaker models for each speaker. Hence, the sub-word grouping technique proposed in Chapter 4 for improving the accuracy of text-constrained speaker recognition merits further research. As the proposed sub-word grouping technique is based on the idea of comparing 'similar' sounds produced by different speakers, it is desirable that a sub-word unit present in two words of different lengths are delineated in an identical manner. However, this cannot be guaranteed by the method of segmenting each word into a pre-selected number of sub-words or by fixing the number of feature vectors in each sub-word. A more flexible segmentation scheme that produces a variable number of variable-sized homogeneous clusters of contiguous feature vectors is likely to yield better results. Realizing such a flexible segmentation scheme without utilizing phonetic transcripts is a challenging task that demands significant innovations to the existing vector quantization and clustering algorithms. Research effort could also be directed towards developing better techniques for grouping the sub-word units.

Word modeling optimization: There is potential for further improving the word recognition accuracy through refinement and optimization of the proposed technique in which each word is modeled as a temporal sequence of GMMs with each GMM representing a sub-word unit. In particular, the effect of time-aligning the utterances used for modeling a word in order to normalize the variations in the utterance lengths is worth evaluating. As the dynamic programming methods typically used for time-alignment are compute-intensive, efficient alternatives may need to be explored. A related task is to evaluate if the use of overlapping sub-word units can render the technique robust to variations in articulation of the same word. It is also beneficial to investigate if using a variable number of sub-word units depending on the word length results in better recognition accuracy.

Compensating background noise and handset variations: Mobile implementations of the proposed speaker-aware word recognition system need to operate in uncontrolled conditions, where background environmental noise and variations in handset characteristics are known to adversely affect the recognition performance. Hence, the study and incorporation of computationally-efficient techniques for countering the effect of environmental noise and handset variations is an important area for future work.

References

[1] Ladefoged, P., “Phonetic Basis or Computer Speech Processing”, *Computer Speech Processing*, Edited by F. Falliside and W.A. Woods, Prentice Hall International, London, 1983.

[2] Flangan, J.L., “Voices of Man and Machine”, *Journal of Acoustic Society of America,* March 1972.

[3] Deller, J.R., Hansel, H.L. John and Proakis, J.G., *Discrete Time Processing of Speech Signals*, IEEE, New York, 2000.

[4] Atal, B.S., “Linear Predictive Coding of Speech”, *Computer Speech Processing*, Edited by Falliside, F. and Woods, W.A., Prentice Hall International, London, 1983.

[5] Campbell, J.P. Jr., “Speaker Recognition: A Tutorial”, *Proceedings of IEEE*, Vol. 85, Issue 9, pp. 1437-1462

[6] Davis, S.B. and Mermelstein, P., “Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Text”, *IEEE Trans. On Acoustics, Speech and Signal Processing*, vol. ASSP 28, pp. 357 – 366

[7] Reynolds, D.A., “Experimental Evaluation of Feature for Robust Speaker Identification”, *IEEE Trans on Speech and Audio Processing*, vol. 2, no. 4, pp. 639 – 643.

[8] Reynolds, D.A., "*A Gaussian Mixture Modeling Approach to Text-Independent Speaker Identification*", Ph.D. Thesis, Georgia Institute of Technology, August 1992

[9] Picone J., “Signal Modeling Techniques in Speech Recognition,” *IEEE Proceedings*, vol.81, pp. 1215-1247, Sept. 1993.

[10] Reynolds, D.A., “Experimental Evaluation of Feature for Robust Speaker Identification”, *IEEE Trans on Speech and Audio Processing*, vol. 2, no. 4, pp. 639 – 643.

[11] Markov, K.P. and Nakagawa S., "Comparison between LPC cepstrum and MFCC for speaker recongition using clean and telephone speech", *Acoustic Society of Japan*, Conference Record, 1-1-17, (1999.3)

[12] Delaney, B., Jayant, N., Hans, M., Simunic, T. and Acquaviva, A., “A Low-Power Fixed Point Front End Feature Extraction for a Distributed Speech Recognition System” *In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP) (2002).*

[13] International Biometric Group, “Template Sizes”, {http://ibgweb.com/reports/public/reports/template_size.html}

[14] Gersho, A., Gray, R., 1991. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, Boston.

[15] Burton, D., 1987. "Text-dependent speaker verification using vector quantization source coding". *IEEE Trans. Acoustics, Speech, Signal Process.* 35 (2), 133–143.

[16] Kinnunen, T., KilpelSinen, T., Fra¨nti, P., 2000. "Comparison of clustering algorithms in speaker identification". *In: Proc. IASTED Internat. Conf. on Signal Processing and Communications (SPC 2000)*, Marbella, Spain, September 2000, pp. 222–227.

[17] Mason, K.Yu.J., Oglesby, J., “Speaker recognition using hidden Markov models, dynamic time warping, and vector quantization,” *Proc. Inst. Elect. Eng., Vis., Image, Signal Process.*, vol. 142, no. 5, pp. 313–318, 1995.

[18] Gupta, H., Hautamki, V., Kinnunen, T. and Frnti, P., *Field Evaluation of Text-Dependent Speaker Recognition in an Access Control Application*. Paper, downloadable at http://cs.joensuu.fi/pages/tkinnu/webpage/pdf/DTWpaper.pdf.

[19] Sakoe, H., Chiba, S., “Dynamic programming algorithm optimization for spoken word recognition”, *IEEE, Trans. Acoustics, Speech, and Signal Proc.*, Vol. ASSP-26, 1978.

[20] Itakura, F., "Minimum prediction residual principle applied to speech recognition", *IEEE Trans. Acoust. Speech and Signal Process.* ASSP-23 (1975) 67–72.

[21] Furui, S., "Cepstral analysis technique for auromatic speaker verification". *IEEE Trans.* ASSP-29, pages 254-272, 1981

[22] Ramasubramanian V., Das A. and Kumar V.P (2006). “Text-dependent speaker recognition using one-pass dynamic programming algorithm”, *In Proc. ICASSP 2006*, vol. 1, pp. 901-904.

[23] Dempster, A., Laird, N., Rubin, D., "Maximum likelihood from incomplete data via the EM algorithm", *J. Royal Statist. Soc. 39* (1977) 1–38.

[24] Moon, T.K., "The expectation-maximization algorithm", *IEEE Signal Proc.* Mag. 13 (1996) 47–60.

[25] Reynolds, D. A. and Rose, R. C., "Robust text-independent speaker identification using Gaussian mixture speaker models", *IEEE Trans. Speech Audio Process.* 3 (1995), 72–83.

[26] Besacier, L., Bonastre, J.-F., 2000. "Subband architecture for automatic speaker recognition". *Signal Process.* 80, 1245–1259.

[27] Besacier, L., Bonastre, J., Fredouille, C., 2000. "Localization and selection of speaker-specific information with statistical modeling". *Speech Comm.* 31, 89–106.

[28] Reynolds, D.A., Quatieri, T.F. and Dunn, R.B., "Speaker Verification Using Adapted Gaussian Mixture Models", *Digital Signal Processing*, (10), 2000, pp. 19-41.

[29] Reynolds, D. A., "Comparison of background normalization methods for text-independent speaker verification". In *Proceedings of the European Conference on Speech Communication and Technology,* September 1997, pp. 963–966.

[30] Panda, A., "*High Performance Voice Authentication System*", M.Engg Thesis, Nanyang Technological University, 2003.

[31] Bocklet, T., and Shriberg, E. "Speaker recognition using syllable-based constraints for cepstral frame selection". In *Proc. Int. conference on acoustics, speech, and signal processing (ICASSP 2009)* (Taipei, Taiwan, April 2009), pp. 4525 – 4528.

[32] Brummer, N., Burget, L., Cernocky, J., Glembek, O., Grezl, F., Karafiat, M., Van Leeuwen, D., Matejka, P., Schwarz, P., and Strasheim, A., "Fusion of heterogeneous speaker recognition systems in the STBU submission for the NIST speaker recognition evaluation 2006," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 7, pp. 2072–2084, 2007.

[33] Fattah, M.A., Ren, F., Kuroiwa, S., "Phoneme based speaker modeling to improve speaker identification", *AIML International Conference*, June 2006

[34] Stapert, R.P., Mason, J.S., "A Segmental Mixture Model for Speaker Recognition", *Proceedings of Eurospeech 2001*, Vol. 4, pp. 2509-2512, 2001

[35] Stapert, R.P., "*A segmental mixture model: maximising data usage with time sequence information*", PhD Thesis, University of Wales Swansea, March 2001.

[36] Rabiner, L.R., Juang, B.H., *Fundamentals of Speech Recognition*, Prentice-Hall, Englewood CliDs, NJ, 1993.

[37] Picone, J., "Continuous speech recognition using hidden Markov models", *IEEE Acoust Speech Signal Process*. Mag. 7 (1990) 26–41.

[38] Rabiner, L.R., "A tutorial on hidden Markov models and selected applications in speech recognition", *Proc. IEEE 77* (1989) 257–286.

[39] Zhu, Y., Gao, S., Ran, F., Chen, I., Macleod, B., Millar & Wagner, M., (1994). "Text-independent speaker recognition using VQ, mixture Gaussian VQ and Ergodic HMMs". *Proceedings* of *the ESCA Workshop on Automatic Speaker Recognition, Identification and Verification,* Martigny.

[40] Matsui, T., and Furui, S., "Comparison of Text-Independent Speaker Recognition Methods Using VQ-Distortion and Discrete/Continuous HMMs," *Proc. Int.Conf. Acoustics, Speech and Signal Processing,* San Fransisco, CA, vol. 2, pp. 157-160, March 1992.

[41] Veth, J., and Bourlard, H., "Comparison of hidden Markov model techniques for automatic speaker verification in real world conditions," *Speech Communication*, vol. 17, Mar 1995, pp. 81–90.

[42] Rosenberg, A. E., Lee, C. H., Gokoen, S., "Connected word talker verification using whole word hidden Markov model," in *ICASSP-91,* 1991, pp. 381–384.

[43] Matsui, T., and Furui, S., "Concatenated phoneme models for text-variable speaker recognition", In *ICASSP-93,* Minneapolis, 1993.

[44] Rosenberg, A. E., Lee, C. H., Soong, F.K., "Sub-Word Unit Talker Verification Using Hidden Markov Models," In *ICASSP-90.*

[45] Euler, S., Langlitz, R., Zinke, J.,"Comparison of whole word and subword modeling techniques for speaker verification with limited training data", In *ICASSP–97*, Munich (Germany), 1997.

[46] Sturim, D.E., Reynolds, D.A., Dunn, R.B. and Quatieri, T.F. "Speaker Verification using Text-Constrained Gaussian Mixture Models", *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pp. 677-680, May 2002.

[47] Boakye, K., "*Text-Constrained Speaker Recognition using Hidden Markov Models*", Final Project Paper, University of California, Berkeley, 2003.

[48] Fortuna, T. F., "Dynamic Programming Algorithms in Speech Recognition", *Informatica Economică nr.* 2(46), pp. 94-99, 2008.

[49] Reynolds, D.A., Rose, R.C., and Smith, J.T., "PC Based TMS320C30 Implementation of the Gaussian Mixture Model Text-Independent Speaker Recognition System", *Proc. Of Int. Conf. Sig. Proc. Applications and Technology*, Nov. 1992 pp. 967-973.

[50] Murthy, P., *VLSI Based Embedded System for Voice Authentication*, M.Eng Thesis, Nanyang Technological University, Nov. 2000.

[51] Przybocki, M. A. and Martin, A. F., 'NIST Speaker Recognition Evaluation - 1997', *Proc. RLA2C* 1998, Avignon, pp120-123.

[52] Deller, J.R., Hansel, H.L. John and Proakis, J.G., *Discrete Time Processing of Speech Signals*, IEEE, New York, 2000.

[53] Huang, C., Tao, C., AND Chang,E., (2004). "Accent Issues in Large Vocabulary Continuous Speech Recognition", *International Journal of Speech Technology*(7): 141-153.

[54] Shore, I. E. and Burton, D., "Discrete utterance speech recognition without time normalization-recent results," *Proceedings of Int, Conf. Pattern Recognition*, pp.582-584, IEEE 82CH1801-0 (Oct. 1982).

[55] Abdulla, W., Chow, D., Sin, G., "Cross-words reference template for DTW-based speech recognition systems", *in Proc. IEEE* TENCON, Bangalore, India, 2003.

[56] Zhang, Y., Desilva, C.J.S., Togneri, A., Alder, M., Attikiouzel, Y., "Speaker-independent isolated word recognition using multiple Hidden Markov Models". In *Proc. IEE Vision, Image and Signal Processing*, volume 141, 3, pages 197–202, June 1994.

[57] Lee, K.F., "Context-dependent phonetic hidden Markov models for speaker-independent continuous speech recognition", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 38, no. 4, pp. 599~609, Apr. 1990

[58] Rabiner, L., Juang, B. H., "*Fundamentals of Speech Recognition*", Prentice-Hall, New Jersey, 1993

[59] Schwartz, R. M., Chow, Y. L., Roucos, S., Krasner, M., Makhoul, J., "Improved hidden Markov modeling of phoneme for continuous speech recognition", *in IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, 1984

[60] Smith, F.J., Ming, J., O'Boyle, P., Irvine, A.D., "A Hidden Markov Model with Optimized Inter-Frame Dependence", *Proc. of ICASSP*, pp. 209-212, 1995.

[61] FSugamura, N., Shikano, K., Furui, S., "Isolated word recognition using phoneme-like templates", *Proc. of ICASSP*, pp. 723-726, 1983.

[62] Rogers, F., "*On the application of vector quantization to speaker independent isolated word recognition*", Masters Thesis, Simon Fraser University, 1996.

[63] Everitt, B.S., Hand, D.J., *Finite Mixture Distributions,* Chapman and Hall, New York, 1981.

[64] Prabhakar, N., *Implementation of Voice Authentication on a 32-bit Micro controller*, FYP Report, Nanyang Technological University, Singapore, 2002.

[65] Singh, G., Panda, A., Bhattacharyya, S., Srikanthan, T., "Vector quantization techniques for GMM based speaker verification". In *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP 2003)*, Hong Kong, 2003.

[66] Furui, S., *Digital Speech Procssing, Synthesis and Recognition*, Marcel Dekker Inc., New York, 1989.

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

[68] Specification of KING Speech Database (http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC95S22)

[69] Specification of TIMIT Speech Database (http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC93S1)

[70] Auckenthaler, R. and Mason, J., "Gaussian Selection Applied to Text-Independent Speaker Verification", *In Proceedings of A Speaker Odyssey - Speaker Recognition Workshop*, 1997.

[71] McLaughlin, J., Reynolds, D. A. and Gleason, T., "A study of computation speed-ups of the GMM-UBM speaker recognition system," *In Proc. 6th European Conf. Speech Communication and Technology (Eurospeech 1999)*, Budapest, Hungary, 1999, pp. 1215–1218.

[72] Beigi, H.S.M., Maes, S.H., Sorensen, J.S., Chaudhari, U.V., "A hierarchical approach to large-scale speaker recognition". *In Proc. 6th European Conference on Speech Communication and Technology (Eurospeech 1999)*, pages 2203–2206, Budapest, Hungary, 1999.

[73] Sun, B., Liu, W., Zhong, Q., "Hierarchical speaker identification using speaker clustering". In *Proc. International Conference on Natural Language Processing and Knowledge Engineering 2003*, pages 299–304, Beijing, China, 2003.

[74] Pellom, B. L. and Hansen, J. H. L., "An efficient scoring algorithm for gaussian mixture model based speaker identification," *IEEE Signal Process. Lett*., vol. 5, no. 11, pp. 281–284, Nov. 1998.

[75] Reynolds D. A. and Heck, L. P., "Integration of Speaker and Speech Recognition Systems", *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 869-872, 1991.

[76] Aronowitz, H., Burshtein, D., Amir, A., "Text Independent Speaker Recognition Using Speaker Dependent Word Spotting". *In Proceedings of the International Conference of Spoken Language Processing (ICSLP '04)*, Jeju Island, South Korea, pp. 1789–1792 (2004)

[77] Park, A. and Hazen, T.J., "ASR Dependent Techniques for Speaker Identification", *In Proceedings of the International Conference on Spoken Language Processing*, 2002.

[78] Chen, W., Zhenjiang, M., Xiao, M., "Differential MFCC and Vector Quantization used for Real-Time Speaker", *IEEE, Congress on Image and Signal Processing*, 978-0-7695-3119-9/08, pp. 320-323, 2008.

[79] Gopinath, R.A., "Maximum Likelihood Modeling with Gaussian Distribution for Classification", *Proc of ICASSP*, 1998, vol. II, pp. 661 – 664.

[80] Tong, Y.L., *The Multivariate Normal Distribution*, Springer Verlag, New York, 1990.

[81] Weber, F., Peskin, B., Newman, M., Corrada-Emmanuel, A., Gillick, L., "Speaker Recognition on Single- and Multispeaker Data," *Digital Signal Processing* 10(1-3): 75-92

[82] SRI's LVCSR Based Speaker Recognition System, *NIST Speaker Recognition Evaluation Workshop*, 1997.

[83] Specification of TI46 Speech Database (http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC93S9)

[84] Navr´atil, J., Chaudhari, U.V., Maes, S.H., "A speech biometrics system with multigrained speaker modeling", *Proc. Conference for Natural Speech Processing (KONVENS2000)*, Ilmenau, Germany, October 2000.

[85] James R. Glass and Victor W. Zue, "Multi-level acoustic segmentation of continuous speech". *In Proceedings of ICASSP,pages 429-432, 1988.*

[86] Svendsen, T. and Soong, F., "On the automatic segmentation of speech signals". *In Proceedings of ICASSP*, pages 3.4.1-3.4.4, 1987.

[87] Juang, B.H. and Rabiner, L.R., "The Segmental K-means algorithm for estimating parameters of Hidden Markov models". *IEEE Trans. on Acoustics, Speech and Signal proc.*, 38(9):1639{1641, September 1990.

[88] Ljolje, A., Riley, M.D., "Automatic segmentation and labeling of speech". *In Proceedings of ICASSP*, pages 473{476, 1991.

[89] Mermelstein, P., "Automatic segmentation of speech into syllabic units". *Journal of Acoustical Society of America*, 58(4):880-883, October 1975.

[90] Lewis, E., "Automatic segmentation of recorded speech into syllables for speech synthesis", *In Proceedings of EuroSpeech '01, Aalborg, pp. 1703-1707.*

[91] Van der Hulst, H. and Ritter, N., "The Syllable: Views and Facts", *Walter de Gruyter*, Berlin, 1999.

[92] Gartner Report on Worldwide Mobile Device Sales in 2010 (http://www.gartner.com/it/page.jsp?id=1543014)

[93] Gartner Report on Mobile Communication Devices Sold Worldwide in First Quarter 2011. (http://www.gartner.com/it/page.jsp?id=1689814)

[94] Android Architecture Documentation (http://developer.android.com/guide/basics/what-is-android.html)

[95] Android MediaRecorder API Documentation (http://developer.android.com/reference/android/media/MediaRecorder.html)

[96] Android AudioRecord API Documentation (http://developer.android.com/reference/android/media/AudioRecord.html)

Appendix A: Conference Paper

The following is the extended abstract of the paper presented in the Second Annual Summit and Conference of the Asia-Pacific Signal and Information Processing Association (APSIPA'10) - Student Symposium, Singapore, December 2010.

SUB-WORD CONSTRAINED GAUSSIAN MIXTURE MODEL FOR ISOLATED WORD RECOGNITION

George Rosario Dhinesh, G.R. Jagadeesh, Ashish Panda and Thambipillai Srikanthan
Centre for High Performance Embedded Systems, Nanyang Technological University
E-mail: {geor0006, asgeorge, pand0005, astsrikan}@ntu.edu.sg

Abstract- We aim to improve the effectiveness of using Gaussian Mixture Model (GMM) for word recognition by incorporating temporal information of speech. A time-ordered sequence of GMMs is used to model each word that is segmented into a sequence of sub-word units. Results show a 5.6% increase in the recognition rate over the baseline GMM approach.

I.INTRODUCTION

State-of-the-art word recognition systems, typically based on Hidden Markov Models (HMM), require high computing power, which is lacked by the embedded devices [1]. GMM-based word recognition can potentially be a computationally-efficient alternative. However, modeling a word with GMM fails in capturing the temporal information of speech because of the bag-of-frames approach it follows to model the statistical distribution of all the speech frames. Hence, in order to make GMM usable for word recognition, there is a need to incorporate time sequence information into it. We present an approach in which each word is modeled as a time ordered GMM sequence, where the sequence corresponds to the order of sub-word units that form the word. This enables us to have a temporally-constrained sub-word level matching for the word recognition problem.

II.SUB-WORD CONSTRAINED GMM

In the proposed approach, each word, represented as a sequence of speech feature vectors, needs to be segmented into multiple sub-word units or segments. Our word recognition technique, which operates purely at the acoustic level, does not require the segments to correspond to any linguistic elements such as phonemes. We evaluate two segmentation methods. The first, simple equal segmentation, segments a given word into n equal parts. The second, constrained clustering segmentation, is an adaptation of the technique proposed in [2]. It segments a word of T feature vectors into n segments, by obtaining a set of segment boundaries t_b, b=1,2...n such that the global distortion measure

$$D = \sum_{b=1}^{n} \sum_{t=t_b}^{t_{b+1}-1} d(y_t, c_b) \qquad (1)$$

is minimized, where y_t is the feature vector for the t^{th} frame, c_b is the centroid of vectors in the b^{th} segment and d is the local distortion. We use the Euclidean distance as the distortion measure.

To model a word, multiple utterances of the word are considered. We utilize the fact that the time order of the sub-word portions will be the same for all these utterances. A GMM is trained for each sub-word by grouping parallel sub-words from all the utterances. During testing, the test utterance is segmented into the same number of sub-words and each sub-word is scored only against the corresponding GMM. The overall likelihood score is obtained by averaging the scores of all the sub-words.

III.RESULTS AND CONCLUSIONS

The proposed technique was evaluated on the TI46 database, which contains multiple utterances of 20 words from 16 speakers. The feature vectors, each with 20 Mel Frequency Cepstral Coefficients, were calculated from a 20 msec speech window progressing at a rate of 10 msec. In the baseline GMM approach, the best recognition rate of 90.6% was achieved when 16 mixtures were used to model each word. In comparison, as shown in Table I, the sub-word constrained GMM approach achieved an accuracy of 96.2% when each word was segmented into 4 segments using the simple equal segmentation method, which is computationally trivial. The results indicate the potential of our approach for isolated word recognition. Also, unlike HMM, our technique does not require initial state and state transition probabilities for preserving the time sequence information and thus nullifies the complex computations involved. This makes our approach suitable for devices with limited computing resources. The robustness could be possibly improved by exploring certain variations of the proposed technique like having different number of segments and mixtures for different words.

Segmentation method	Word recognition rate (%)			
	Segments / word = 2	**Segments / word = 3**	**Segments / word = 4**	**Segments / word = 5**
Constrained clustering segmentation	89.61	94.10	95.06	93.94
Simple equal segmentation	95.28	95.69	96.26	95.49

TABLE I: SUB-WORD CONSTRAINED GMM FOR WORD RECOGNITION

REFERENCES

[1] Christophe L´evy, Georges Linar`es, Jean-Franc¸ois Bonastre, "GMM Based AcousticModeling for Embedded Speech Recognition" In Proceedings of ICSLP'2006, Pittsburgh, USA, 2006.

[2] T. Svendsen and F. Soong, "On the automatic segmentation of speech signals". In Proceedings of ICASSP, pp. 3.4.1-3.4.4, 1987.