COMBINING SPEECH RECOGNITION AND SPEAKER VERIFICATION

BY AANCHAN K MOHAN

A thesis submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Professor Lawrence R. Rabiner

and approved by

New Brunswick, New Jersey

October, 2008

ABSTRACT OF THE THESIS

Combining Speech Recognition and Speaker Verification

by Aanchan K Mohan

Thesis Director: Professor Lawrence R. Rabiner

Traditional fixed pass-phrase or text-dependent speaker verification systems are vulnerable to replay or spoofing attacks. Random pass-phrase generation, speech verification and text-independent speaker verification could be combined to create a composite speaker verification system, robust to this spoofing problem. This thesis deals with combining speech verification with text-independent speaker verification for this purpose. A method to perform robust, automatic speech verification using a speech recognizer in a forced alignment mode is proposed and evaluated. A text-independent speaker verification system was developed in MATLAB for training and evaluating Gaussian mixture density-based, target speaker and background speaker models. Equal-error rate is the performance metric used in all speaker verification evaluations. To speed up background model training, a simple technique based on sub-sampling or decimating speech frames is presented. Evaluation of two different feature extraction implementations along with an evaluation of the impact on performance of different configurations of the speech features is also carried out. Further, to mitigate problems with reduced training data and to improve performance, Bayesian adaptation of background speaker models with target speaker training data is used to create target speaker models. The performance of these models is evaluated and compared with conventional target speaker models. The impact of the length of test-utterances, variance limiting and the use of training data from multiple recording sessions has also been investigated.

Acknowledgements

Working on this thesis has been a fantastic and transforming experience. I cannot say enough to thank both my advisors Prof.Rabiner and Prof.Rosenberg. While working on this project, they treated me as an equal, gave high priority to the problems I faced and provided me with constant support. I was allowed to make mistakes, ask questions and encouraged to learn. Their zeal for research and insight into solving problems are truly remarkable. The experience of working with them has been extremely rewarding and enriching, both personally and professionally.

Additionally, I want to acknowledge the assistance I received from Prof.Sorin Dusan during the earlier stages of working on my thesis. I would like to thank Prof.Richard Stern and his colleagues at Carnegie Mellon University who supported us at Rutgers with setting up the Sphinx Speech Recognition system. I acknowledge Saket Anand who did a lot of the initial work and experiments with the Sphinx system, and was vital to get me started off. I would like to thank Bill Kish at the CAIP Center, Rutgers for doing his best to help me out with the computing resources and the lab space I needed.

My parents Mr.Seetharaman Mohan and Mrs.Meena Mohan have sacrificed a lot in their lives to help me get this far, and for this I would like to thank them. My sister Srividya Mohan, a fellow graduate student, has been a source of constant inspiration and a fantastic mentor. Lastly, I would like to thank my friends Balwinder Sachdev, Sumit Satarkar and Rohini Pangrikar for their help and support.

Dedication

To my family.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Problem Definition

Speaker recognition is a term that is collectively used to describe the task of either automatically identifying a speaker from a sample of speech provided by the speaker or verifying an identity claim made by a speaker against a sample of speech provided by the speaker. The former task is called speaker identification and the latter task is called speaker verification. This thesis concerns itself with the task of speaker verification in the context of its application in voice-based authentication.

A speaker verification system is "trained" to "learn" certain characteristics that are unique to a speaker's voice from a sample(or samples) of speech that is(are) specifically and willingly provided by the speaker during a formal *enrollment* session. The characterization of the speaker's voice that the system obtains through the process of "training" is commonly referred to as a speaker model. Individuals who willingly *enroll* themselves into the speaker verification system are called *customers*. Voice-based authentication systems typically employ two-factor authentication by combining the use of speaker verification with some sort of personal identification such as a PIN, which allows a user to make an identity claim. A customer's speaker model is then directly associated with this form of personal identification forming a one-to-one mapping between the two.

Once enrolled, a user, would want to subsequently access certain services or information that is protected by a voice-based authentication system. For this the user must obtain the appropriate authorization from the system. This happens during an *authorization session*. Initially, the user makes an identity claim using the said piece of personal information. Further, the user is required to say a *pass-phrase*, into the speaker verification system, thereby providing a speech sample. The speech sample thus obtained is then *tested* by the verification system against the identity claim made.

Depending on the kind of authentication system, the content and the nature of the pass-phrase could remain fixed, requiring it to be repeated during each authentication session by the user. Or, the user may be prompted to say a different pass-phrase during each authentication session. Fixed pass-phrase systems are thus those systems in which the text of the speech samples provided during the enrollment and authentication sessions is identical. Such systems are commonly referred to as *text-dependent* speaker verification systems. There exists a second category of speaker verification systems in which there is absolutely no constraint placed on the text of the pass-phrase. Thus the text of the speech samples, provided during the enrollment and authentication sessions are independent of each other. Systems in this second category are called *text-independent* speaker verification systems.

Any authentication system may suffer attacks from malicious individuals who intentionally want to break through a security system. Such individuals are commonly referred to as *impostors*. Impostors are people who tend to pose as valid customers. A possible attack that voice-based authentication systems can suffer is what is known as a *replay* or a *spoofing* attack. In such an attack, during an authorization session, an impostor first makes an identity claim by posing to be a valid customer. The impostor then replays pre-recorded speech actually spoken by the customer into the speaker verification system in order to gain access through the system. Such an attack is especially dangerous in a fixed pass-phrase voice authentication system where the pass-phrase does not change from one authentication session to the next.

Fixed pass-phrase or text-dependent systems provide good authentication performance and have been commercially deployed. As stated in [5], one of their major drawbacks though is that they are vulnerable to replay attacks, such as the one described previously. Thus there is certainly a need for voice-based authentication systems that are robust to such attacks. Such robustness could be obtained by the use of a verification system that uses randomized pass-phrases. Such a system could be realized using *text-independent* rather than *text-dependent* processing provided there is an independent process for verifying the text of the input speech sample. This thesis thus concerns itself with the development of a *text-independent* speaker verification system which includes spoken text verification.

The idea around developing such a text-independent system is that during the authentication session instead of using a fixed pass-phrase, the user would be prompted to say a short 5-7 word *unique* system-specified pass-phrase after making an identity claim. The nature of this pass-phrase, in terms of its text, should then be unique to each authentication session or in other words it should be truly random. In such a paradigm, after an identity claim is made and a speech sample is provided by saying a randomly generated system-specified pass-phrase, the authentication system would then need to,

1. verify that the text of the speech sample provided by the user matches the text of the system specified pass-phrase. This is called *speech verification.* The goal of performing speech verification is to essentially overcome any spoofing attempt that uses pre-recorded speech.

2. verify the speaker's identity claim by "testing" the speech sample against the claimed customer's speaker model, once the text of the speech sample matches the text of the pass-phrase. Since the nature of the text is not known during the process of enrollment a text-independent speaker verification system needs to be used.

Thus the main goal of this thesis is naturally subdivided into separately studying the best way to perform both of the above tasks, so that they can be combined to provide a novel and a more robust method for performing speaker verification.

1.2 A Description of the Dynamic Pass-phrase Voice Security System

Figure 1.1 gives an overall view of a voice authentication system, whose need was motivated in Section 1.1. This system is called a Dynamic Pass-phrase Verification System. The figure shows both the *enrollment* and the *authentication* phases.

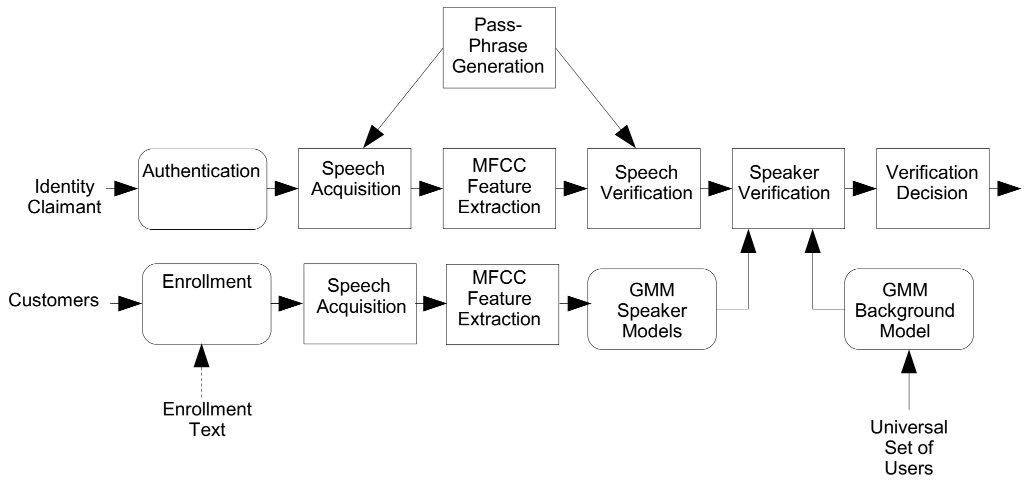


Figure 1.1: The Dynamic Pass-phrase Verification System

In the *enrollment* phase legitimate users or *customers* provide speech samples for the purpose of "training" the system. The content of these training samples could be read text. In the figure we refer to the text used in the enrollment session as *enrollment text*. The "training" speech for each customer is acquired using a *speech acquisition* device, which could be a microphone or a telephone handset. This acquired speech is then passed through the *feature extraction* module to extract features which provide a parsimonious representation of the speech. Mel-Frequency Cepstral Coefficients(MFCCs) [6] and their derivative approximations are used as features in this system. These features are then used to "train" a speaker dependent model which aims at statistically characterizing the enrolling customer's voice. A second set of models called Background Models[7], intended to statistically characterize the global set of speakers is also created. The training of these Background Models is done by pooling speech from a large number of speakers not including the customers. The customer models and background models use statistical representations called Gaussian mixture probability density functions [7, 8, 9] to obtain a text-independent characterization of the statistical properties of voices of specific customers in the customer speaker models and a global representation in the background model.

During the *authentication* phase a user first makes an identity claim by using a PIN or an account number. The *identity claimant* is then prompted to utter a randomly generated pass-phrase of length 5-7 words, generated by the *pass-phrase generation* module. After acquiring this speech utterance, the system then extracts the sets of MFCC feature vectors. Using these extracted features, first the *speech verification* module tries to verify that the claimant did say the required system-specified pass-phrase. If this verification is positive then the *speaker verification* module compares the features extracted from the voice sample against the identity claimant's GMM(Gaussian mixture speaker model) and his/her Background GMM model to obtain a confidence score. Depending on whether this score is above or below a system and user specific threshold, the *verification* module either accepts or rejects the identity claim. Two types of errors can occur. The system might falsely reject the identity claim of a legitimate customer, or the system might falsely accept the speaker identity claim of an impostor. For the purpose of evaluating performance the threshold score is set experimentally for each customer to equalize these two error rates. We call this threshold the equal-error rate threshold.

1.3 Real World Applications

The applications of voice-based authentication systems, such as the one that forms the subject of study of this thesis, are numerous and varied. The list below, discusses some of the possible application areas:

- Access Control: To control access to physical facilities which need a high level of security and restricted access. For additional security, voice-based authentication systems could also be used in combination with other biometric personal identification systems such as fingerprints and retinal scans. Additional applications could include access to personal computers, access to computer networks or personal access to ones own car, or allowing restricted access to certain documents available electronically(document authorization).

- Transaction Authentication: In the financial services industry, robust voice-based authentication is an attractive option to carry out sensitive transactions over the telephone or other communication networks. Telephone banking is a good example of such an application of this technology.

Chapter 2

Speech Verification

2.1 What is Speech Verification?

Section 1.1 suggested a way to combat spoofing of current text-dependent voice authentication systems, and section 1.2 described the outline of the Dynamic Pass-phrase System which aims at combatting this problem of spoofing through the use of a random pass-phrase generation module.

The idea is that an identity claimant I who claims to be customer S, is prompted by the system to say a random pass-phrase with text P. It is assumed that I will say the pass-phrase, by providing a voice sample $\mathbf{X}$ within a certain system specified time-out period. To ascertain that I did say the required pass-phrase, the linguistic content of the speech signal $\mathbf{X}$ must match the text P. This is basically the goal of speech verification. If I tries to spoof the system with pre-recorded speech then there is a very high likelihood that the linguistic content of $\mathbf{X}$ will not match P. Such a scenario calls for a speech verification module which can reliably and accurately overcome this spoofing attempt by I on the basis of the mismatching text of the pass-phrases. If the attempt is overcome at this stage then there is no need to go on to the speaker verification stage where the identity claim of I being S needs to be verified on the basis of the voice utterance $\mathbf{X}$. The speech verification and the random pass-phrase generation module essentially stand guard to reduce the chances of malicious impostors exploiting the vulnerabilities in the speaker verification system.

2.2 The proposed method for Speech Verification

In this thesis we propose a new method for speech verification using a speech recognition engine in a *forced alignment* mode.

2.2.1 Forced Alignment Speech Recognition

A speech recognizer is a search engine which tries to recognize an incoming speech signal and convert it into text. It does so by the use of a set of *acoustic models* for the phonemes (the basic language sounds), a *language model* which describes the syntax of valid sentences and a dictionary of words that it will encounter in the context of the application for which the recognizer is used. This set of models is also known as a speech recognizer's knowledge base.

An acoustic model consists of a set of statistical representations of the basic sounds (phonemes) of speech that make up each word. A language model on the other hand determines the probabilities of sequences of words occurring in a language. Given a speech utterance $\mathbf{X}$, a speech recognizer, using its existing knowledge base tries to find the most probable word string $W^{'}$ that best fits the given utterance in a probabilistic sense, an acoustic sense and a linguistic sense. To better understand this process, we review the probabilistic formulation of the speech recognition problem.

First, let $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^{T}$ represent a set of speech feature vectors and $W = \{\mathbf{w}_i\}_{i=1}^{N}$ represent a string of words of length N which come from a vocabulary V. Then, if $P(W|\mathbf{X})$ is used to denote the probability that the words W were spoken, given that the evidence $\mathbf{X}$ was observed, the recognizer attempts to find the word string $W^{'}$ the maximizes the probability of the word string given the set of speech feature vectors, i.e.,

$$W^{'} = \arg\max_{W} P(W|\mathbf{X}) \tag{2.1}$$

With good quality spoken input and a good recognizer, the word string $W^{'}$ that provides the highest probability score is the actual spoken word string or a close approximation. A speech recognizer tries to maximize this probability $P(W|\mathbf{X})$, while performing its

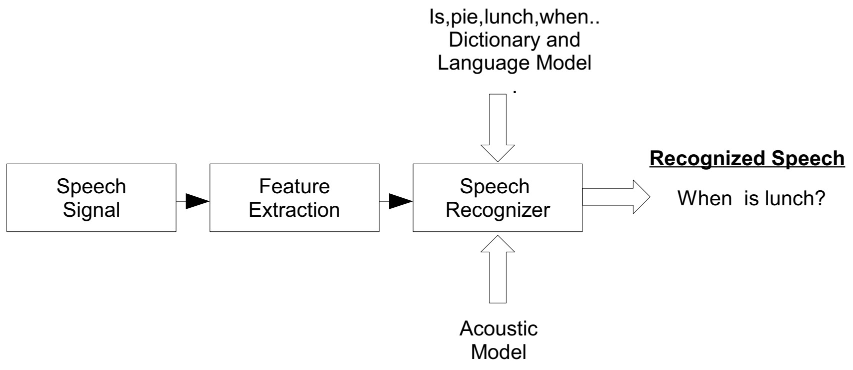


Figure 2.1: The Speech Recognition Process (After The Institute of Signal and Information Processing [1].)

search for the most probable word string $W^{'}$. Based on Bayes' rule the probability to be maximized can be simplified to the form,

$$P(W^{'}) = \max_{W} P(\mathbf{X}|W)P(W) \tag{2.2}$$

where $P(W)$ is called the *likelihood* of the word string W and $P(\mathbf{X}|W)$ is called the *likelihood* of the observation $\mathbf{X}$ for a particular word string W. $P(W)$ forms the language model component of the total likelihood score and $P(\mathbf{X}|W)$ forms the acoustic model component of the total likelihood score. For simplicity we denote the acoustic model log-likelihood score by A and the language model log-likelihood score by L. In the log-domain then the total log-likelihood score is obtained by adding A and L. The total log-likelihood score is thus a measure which helps one quantify how good a fit the recognition hypothesis W is for the given speech utterance $\mathbf{X}$. Fig. 2.1 illustrates the speech recognition process.

Acoustic models of phonemes are obtained by a process of "training", using a database of recordings of speech utterances along with their corresponding text transcriptions. This training process usually bootstraps itself from an initial set of acoustic models(often obtained from a completely different context) and iteratively improves over multiple passes with the training data. Essentially the training process iteratively determines the best alignment of the training speech with the set of phonemes corresponding to the spoken inputs, and then builds up models for each of the phonemes based on the current alignments with the training speech.

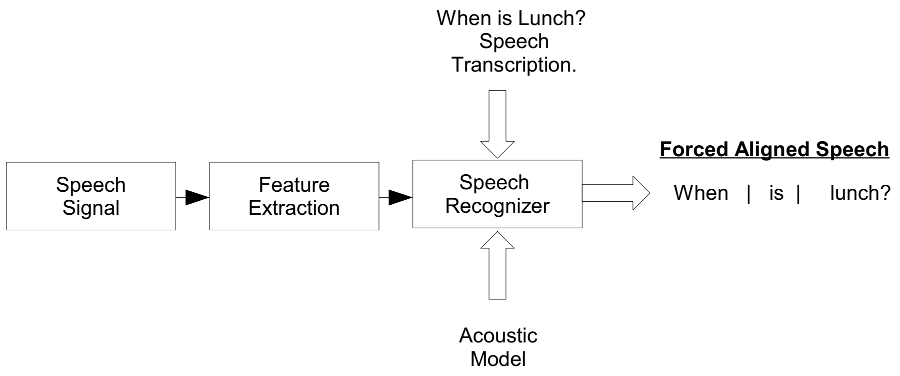


Figure 2.2: The Forced Alignment Process, (After The Institute of Signal and Information Processing [1].)

Figure 2.2 illustrates the forced alignment process and shows how it is different from the process of (unconstrained)speech recognition in Figure 2.1. In contrast to the process of speech recognition, where the recognizer has a language model and a dictionary to work with, in the forced alignment process the speech recognizer is given a speech utterance $\mathbf{X}$ and its exact transcription. As mentioned in [1], using the set of acoustic models for the phonemes of the language, the system then aligns the given transcription with the speech utterance $\mathbf{X}$, identifying which time segments in the speech utterance correspond to individual words in the transcription. During the process of forced alignment, the recognition engine also provides a probabilistic score, called the acoustic likelihood score(since only the acoustic model is used during the forced alignment process), for each word in the transcription, and a total acoustic likelihood score for all the words in the transcription(i.e., the spoken sentence). Henceforth we shall refer to the term *acoustic score* to mean the total acoustic score of an utterance which we denote by $\hat{A}$. This acoustic score can help in quantifying how well a given transcription matches a recorded speech utterance $\mathbf{X}$ or vice-versa. We utilize this system capability in our use of a speech recognizer in the forced alignment mode as a means of speech verification.

2.2.2 Proposed method for Speech Verification

As briefly mentioned in Section 2.2.1 during the general *recognition* process the speech recognizer uses a trained acoustic model and a language model. Therefore for an unknown speech utterance $\mathbf{X}$ the speech recognizer searches for the optimum string of words, i.e., the string that is valid within the constraints of the vocabulary and language model of the speech recognizer, and that has the highest log-likelihood matching score among all valid sentences. We call this recognized string the best *unconstrained* string. On the other hand as mentioned in section 2.2.1, during the *forced alignment* process the speech recognizer performs a highly *constrained* search using only the acoustic models that exactly match the given input string of words.

In the context of the Dynamic Pass-phrase System that was talked about in Section 1.2, we assume that the pass-phrase generation module generates a pass-phrase with text P. Once the prompted identity claimant I says the pass-phrase with text P, by providing a speech utterance $\mathbf{X}$, the speech recognizer first operates in an unconstrained mode to recognize the linguistic content of $\mathbf{X}$ resulting in a word string $W^{'}$ and acoustic score $A^{'}$ and language model score $L^{'}$. The recognition engine is then run in a forced alignment mode, thereby optimally aligning the text of the pass-phrase P with the speech utterance $\mathbf{X}$ to give an acoustic score $\hat{A}$. We would normally expect the unconstrained acoustic score $A^{'}$, for the best(recognized) word sequence, $W^{'}$, to almost always be identical to the recognition score for the constrained input, $\hat{A}$ with the actual spoken word sequence, W. The only cases when these acoustic scores would differ are,

1. The recognizer makes an unconstrained recognition error whereby a different score $W^{''}$ gets a better acoustic score than the true sentence $W^{'}$. Even, when this type of recognition error occurs, the acoustic scores for $W^{'}$ and $W^{''}$ are generally almost identical and thus the difference between $A^{'}$ and $A^{''}$ is small.

2. The spoken sentence doesn't match the one requested by the pass-phrase system in which case the difference between $A^{'}$ and $\hat{A}$ is relatively large and is a very strong indication that an incorrect sentence was spoken.

In summary, if I spoke the text of the pass-phrase P correctly then the difference between $A^{'}$ and $\hat{A}$ should be close to zero. If, on the other hand I turns out to be an impostor trying to spoof the system and speaks a different utterance, then the difference between A and $\hat{A}$ should be very large. In chapter 7, we verify experimentally that the difference score between $A^{'}$ and $\hat{A}$ can be used as a reliable, robust and accurate metric to verify (or reject) the incoming speech sample $\mathbf{X}$.

Chapter 3

Speaker Verification:An Overview

3.1 The Bases for Speaker Recognition

Humans use several perceptual cues to distinguish between speakers. There are high level perceptual cues such as accent, diction, idiolect, pronunciation and various speaker idiosyncrasies. Middle level perceptual cues include prosodics, speech rhythm, intonation and volume modulation. Low-level cues such as the acoustic properties of speech, which are due to the spectral differences associated with the size and shape of an individual's speech articulators, tend to capture the physical traits of each individual. Though there are no exclusive speaker identity cues, low-level acoustic cues are most commonly used in automatic speaker recognition systems because they are the easiest to extract. To understand how these low-level acoustic cues could be captured, it is important to understand the physiological processes that underlie human speech production.

The human speech apparatus, illustrated in Figure 3.1, consists of the vocal tract,

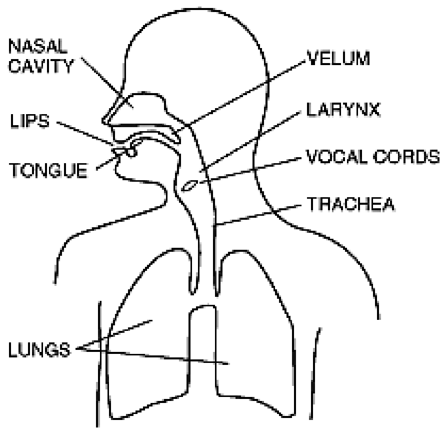


Figure 3.1: The Human Vocal Tract. (After Cummins [2].)

the nasal tract, the vocal cords or the glottis and the sub-glottal system which consists of the lungs and the diaphragm. The vocal tract, begins at the glottis, includes the mouth or the oral cavity and ends at the lips. The nasal tract, extends from the velum(or the soft palate) to the nostrils. The nasal tract acoustically couples to the vocal tract when the velum is lowered. The vocal cords, while interacting with the sub-glottal system, periodically contract and relax to release puffs of air, into the vocal tract. As mentioned in [10], the configuration of the vocal tract(which can be thought of as a configuration of concatenated tubes) then imposes its resonance structure upon this excitation(the periodic puffs of air) so as to produce the different sounds of voiced speech. Unvoiced speech is produced either due to the rushing of air though a constriction in the vocal tract or due to a sudden release of air after a complete closure of a part of the vocal tract. The process of human speech production is illustrated in Fig. 3.2(a).

For voiced sounds, the resonances imposed upon the voiced excitation by the vocal tract appear as relative peaks in the speech spectrum. These spectral peaks associated with the periodic resonances are referred to as speech formants. Further, as stated in [11], the locations in frequency, and to a lesser degree the shapes of the resonances help to distinguish one speech sound from another. This is illustrated in Fig. 3.3. The temporal evolution of various vocal tract configurations, for the production of various speech sounds, occurs rather slowly as time progresses. Vocal tract configurations can thus be modeled by a slowly time-varying linear filter which captures the shape and location of speech formants at various instances in time. Fig. 3.2(b) illustrates the human vocal tract as a source filter model. Excitation for voiced sounds is produced by a periodic pulse generator and excitation for unvoiced sounds is produced by a random noise generator. Fig. 3.2(c) completes the picture by illustrating, in the frequency domain, how the vocal tract shapes the glottal spectrum to produce the speech spectrum.

Further, as stated in [11]:

> Formant locations and bandwidths and spectral differences, associated with the overall size of the vocal tract, serve to distinguish the same sounds spoken by different speakers. The shape of the nasal tract which determines the quality of nasal sounds, also varies significantly from speaker to speaker. The mass of the glottis is associated with the basic fundamental frequency for voiced speech sounds. The average basic fundamental frequency is approximately 100 Hz for adult males, 200 Hz for adult females and 300 Hz for children. It also varies

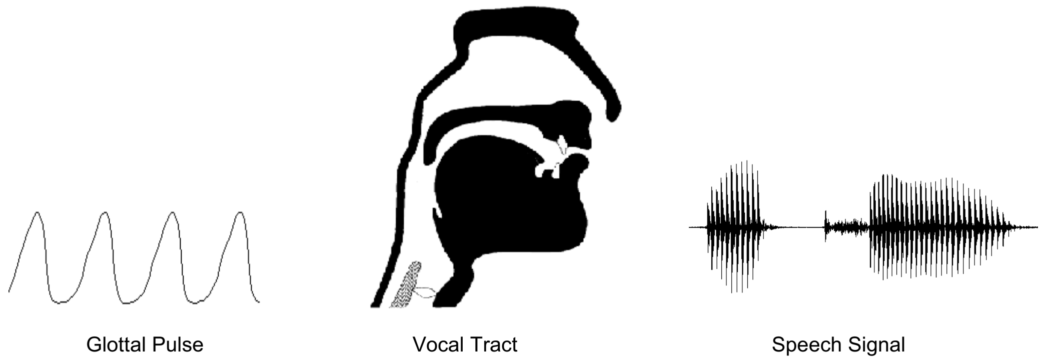


Fig (a)

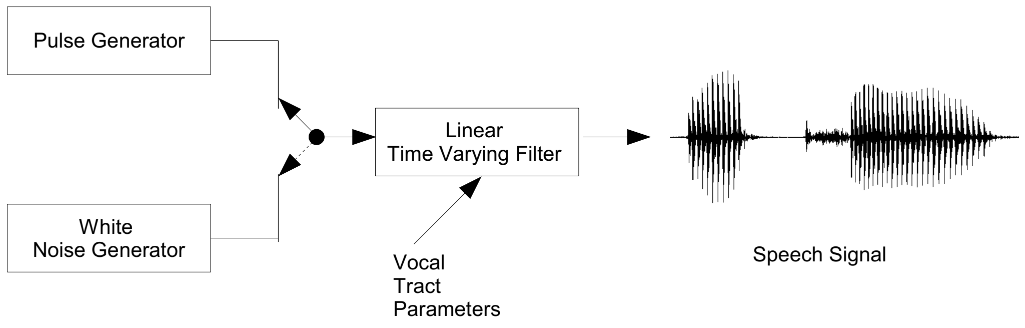


Fig (b)

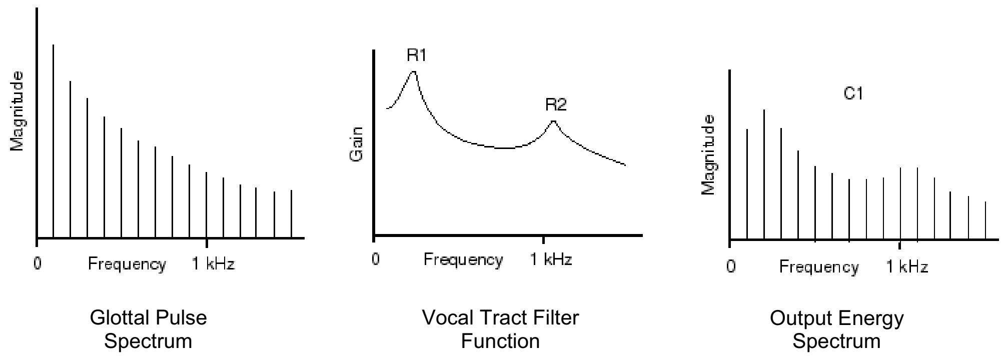


Fig (c)

Figure 3.2: Human Speech Production

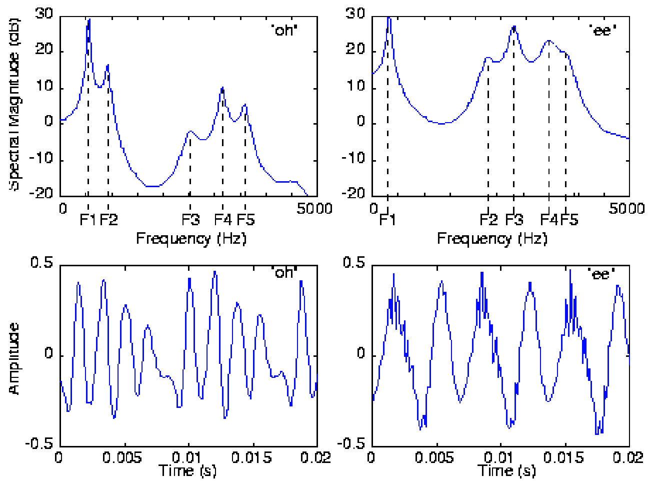


Figure 3.3: Voiced speech sounds "oh" and "ee" with their respective speech waveforms and speech spectra. It is illustrative to see how formant locations help in distinguishing these two voiced speech sounds. (After Johnson [3].)

from individual to individual. This in essence allows the speech signal to capture certain traits specific to a speaker, and hence finds its use as a biometric for personal identification and verification.

It should be noted that pitch information is rarely employed in automatic speaker recognition, for reasons that are mentioned a little later in this thesis.

Spectrograms in Fig. 3.4 illustrate how different the vocal tract configurations can be when two different people utter the same speech sounds.

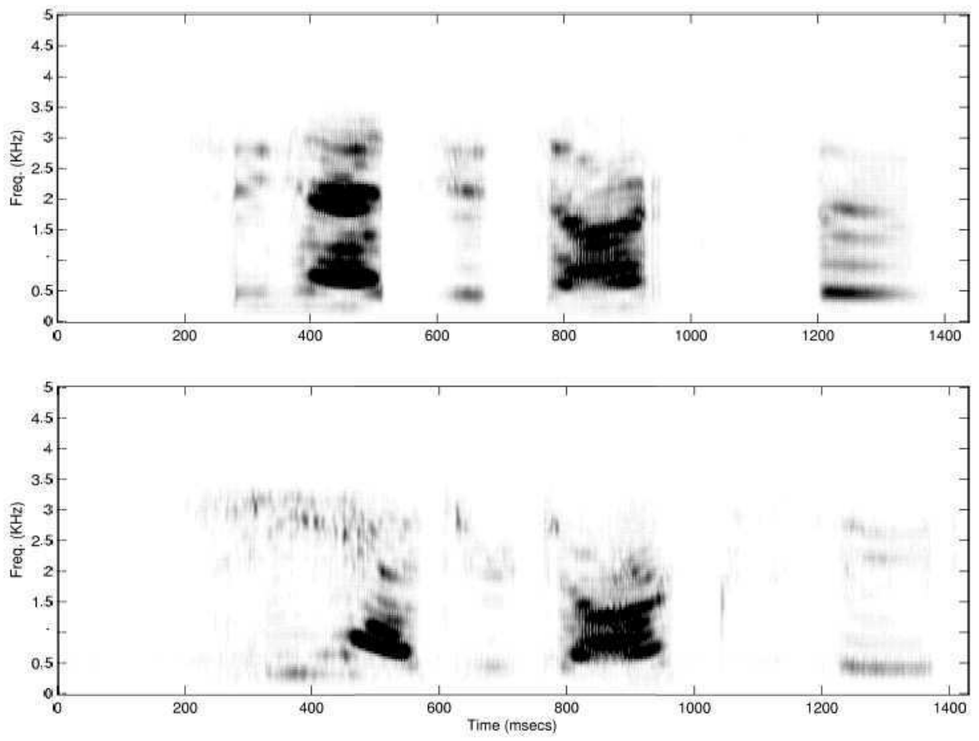


Figure 3.4: Spectrograms with the formants enhanced of two different female speakers saying the utterance “She had your dark suit”

3.2 What is Speaker Verification?

Speaker recognition is a term that usually encompasses the two different tasks of *speaker identification* and *speaker verification.* In the *speaker identification* task a voice sample from an unknown speaker is compared with a set of labeled speaker models. When the set of speaker models includes all of the speakers of interest, the identification task is referred to as a *closed-set* identification task. Therefore if there are N valid labeled speaker models, a closed-set identification system determines the single closest match for the incoming unknown voice sample from the set of N labeled models. On the other hand when the task involves determining whether the speaker is who he/she claims to be then the task is called a *speaker verification* task. This task is also commonly referred to as an *open-set* task since the set of impostors(those falsely claiming to be a valid user) is generally not known to the system. In this case the open-set task involves making a binary(yes/no) decision. One can also merge the two tasks by adding a "none-of-the-above" option to the closed-set identification task, to create another task called the open-set identification task, which is necessary only when there is a possibility of a match to a speaker outside the set of known speakers. As mentioned in Section 1.1 the speaker verification task, in general, could also be classified as either text-dependent or text-independent.

3.3 Approaches to Speaker Recognition

Speaker recognition methods chiefly differ in the manner in which the true speakers or customers are modeled. Depending on the method of modeling the speaker, a method can be broadly classified as either a *parametric* or a *non-parametric* method of speaker modeling. In non-parametric approaches to speaker modeling, which tend to be employed in text-dependent systems, no formal structural model is assumed for characterizing a valid speaker's voice. Parametric approaches, on the other hand, make structural assumptions about the speaker model. As such, in text-independent systems, less-detailed models (parametric models) that model the overall acoustic space of the user's utterances tend to be effective. In [11] it is mentioned that,

> Nonparametric models make few structural assumptions and are effective when there is sufficient enrollment data that is matched to the test data. Parametric models allow a parsimonious representation of the structural constraints and can make effective use of the enrollment data if the constraints are appropriately chosen.

Template-matching and Nearest-Neighbour modeling are examples of non-parametric approaches to speaker modeling. In the template-matching approach [12], all of the feature vectors extracted during the enrolment phase are retained to serve as templates of the customer's voice. Dynamic time-warping is then used to align and measure the similarity between the test speech vectors and the retained voice templates to obtain a match score during the verification process [5]. In the Nearest Neighbour approach [13], akin to the template-matching approach, the speech feature vectors from the enrollment session are stored to serve as the reference feature vectors of the enrolled speaker. During the testing phase, the match score between the test feature vectors and reference features are calculated as the cumulated distance of each test feature to its k nearest neighbours in the set of reference vectors [5].

Among parametric approaches, discriminative modeling techniques such as neural networks(NNs) [14, 15, 16] and more recently support vector machines(SVMs) [17, 18] have been used in speaker recognition tasks. In these techniques speaker models are trained to distinguish a particular speaker from a set of alternative speakers. This lends them to be applied in both the speaker recognition and verification tasks, but re-training these discriminative models, especially NNs could be difficult and cumbersome [5]. Further, quantized vector codebooks, designed using appropriate Vector Quantization(VQ) algorithms [19, 20], on the other hand seek to represent a speaker using a codebook of vectors [21], trained on a speaker's enrollment data. VQ codebooks in [21] were very successfully applied to the text-dependant speaker identification task. Additionally, statistical methods seek to characterize the speaker's voice using probability density functions and/or probabilistic models. In [22], a uni-modal Gaussian probability distribution function is used to model the characteristics of a speaker's voice in a text-independent speaker identification task. Probabilistic models such as the Hidden Markov Model(HMM) [23, 24] have also been used speaker recognition tasks in

[25, 26, 27, 28]. Probability density functions consisting of a mixture of Gaussians, also called a Gaussian Mixture Model(GMM), have been studied extensively in [7, 8] for the purpose of modeling a speaker's voice in the text-independent speaker recognition task.

3.4 Motivation for the use of Probabilistic Methods in Speaker Verification

Probabilistic methods, especially the HMM and the GMM are the dominant techniques for modeling speakers in both text-dependent and text-independent speaker recognition tasks. These methods are mathematically well understood, lend themselves to efficient implementation in hardware and software, are generalizable to new data, and have been shown to give the best performance in noisy environments with appropriate signal processing and normalization techniques. HMMs tend to give better performance in text-dependent tasks since they tend to model temporal transitions between basic speech units well. In the text-independent task, on the other hand, the number of temporal transitions that need to be captured are too many and thus, GMMs(degenerate one-state HMMs) which are thought to model the acoustic space of the units of spoken language, indeed provide the best performance. In the speaker identification task, probabilistic modeling allows the use of the Bayes' rule to arrive at a decision to determine the most probable speaker among a closed-set of N speakers(eg. [9]). Further probabilistic methods are useful since they allow a statistical characterization for the speaker or customer in the form of a speaker model and a separate statistical characterization for a set of alternative speakers or impostors in the form of an anti-speaker or a background model. This allows probabilistic models to be used in a likelihood-ratio detector framework for the speaker verification task (e.g., [7, 29]). Since this thesis concerns itself with the text-independent speaker verification task, we shall discuss Gaussian Mixture Models in this context in greater detail in a later chapter.

Chapter 4

Feature Extraction

4.1 The nature of features used and the analysis of speech

Section 3.1 spoke about low-level perceptual cues that encode the physical traits of each individual. As stated in [30], among these, the pitch information, though it reflects the frequency of excitation from the glottis, is seldom used in speaker recognition since it is strongly affected by factors other than anatomical structure, such as a speaker's emotional state and speech effort. Also, as discussed in Section 3.1, the speech spectrum tends to capture many useful lower-level perceptual cues.

Representations of the speech signal based on the short-term speech spectrum capturing low-level acoustic cues, have proven extremely useful in speaker recognition. As stated in [8],

> Common spectrum representations are linear predictive coefficients(LPC) and their various transformations(reflection coefficients, PARCOR coefficients, cepstral coefficients etc) [10], and filter bank energies and their cepstral representation [6]. One drawback of the LPC representation is that it is based on an all-pole filter model which may omit significant spectral characteristics of speech contaminated with noise. The filterbank energies, however are direct measurements of the energy in different frequency bands and are not subject to any model constraints. Furthermore, the center frequencies and bandwidths of the filters can be adjusted to match the critical bands of the ear so that the filter energies better capture the perceptually important characteristics of the speech signal [6]. The critical band or mel-scale filter bank energies and their cepstral transformation(better known as Mel-Frequency Cepstral Coefficients or MFCCs), thus serve as features that are suited to speaker recognition tasks.

Also, as stated in Section 3.1, the temporal evolution of various vocal tract configurations for the production of various speech sounds, occurs rather slowly with time. As a result the vocal tract properties can be assumed to be stationary for small windows of time(approx. 20-30ms), while slowly changing to a new configuration as time progresses. For this reason, Short-time Fourier Analysis of speech, is often done over small “sliding” time windows called speech frames. Each successive frame is allowed some degree of overlap(between 50-90%) with the previous frame. For speech sampled at 8KHz, as in the MFCC implementations studied in this thesis, analysis windows of size 20ms, with a delay of 10ms between the start of successive frames(thereby allowing a 50% overlap between successive frames), were used.

For telephone speech, the quality of the speech signal often gets degraded by the characteristics of the telephone channel. The telephone channel, is often modeled as a linear time-invariant filter with impulse response $h[n]$. For the nth mel-cepstral feature vector $\{\mathbf{x}_n\}_{n=1}^{T}$, the effect of the filter appears as an additive component [8] as,

$$\mathbf{z}_n = \mathbf{x}_n + \mathbf{h}, \tag{4.1}$$

where $\mathbf{z}$ is an observed cepstral vector, $\mathbf{x}$ is the input speech cepstral vector and $\mathbf{h}$ is the channel filter cepstral vector. In order to minimize the channel filter effects, as stated in [9], it is important to use “channel invariant features”, such as cepstrum difference coefficients. First order difference coefficients, taken $\pm W_1$ frames apart, are calculated as,

$$\begin{aligned} \Delta\mathbf{z}_n &= \mathbf{z}_n - \mathbf{z}_{n-W_1} && (4.2) \\ &= \mathbf{x}_n - \mathbf{x}_{n-W_1}. && (4.3) \end{aligned}$$

The first order difference seeks to remove the fixed transmission factor $\mathbf{h}$ introduced by the channel filter. In addition the second order difference is often calculated in a similar manner over the first order difference frames taken $\pm W_2$ frames apart, i.e.,

$$\Delta^2\mathbf{x}_n = \Delta\Delta\mathbf{x}_n = \Delta\mathbf{x}_n - \Delta\mathbf{x}_{n-W_2}. \tag{4.4}$$

The first and second order difference coefficients thus computed, are simply concatenated to the static MFCC feature vector to give a feature vector with added dimensionality.

Additionally, the first and second order difference coefficients, incorporate dynamic information regarding the variation of the static features or the MFCCs over time. The difference coefficients have been shown to contain speaker specific information and to

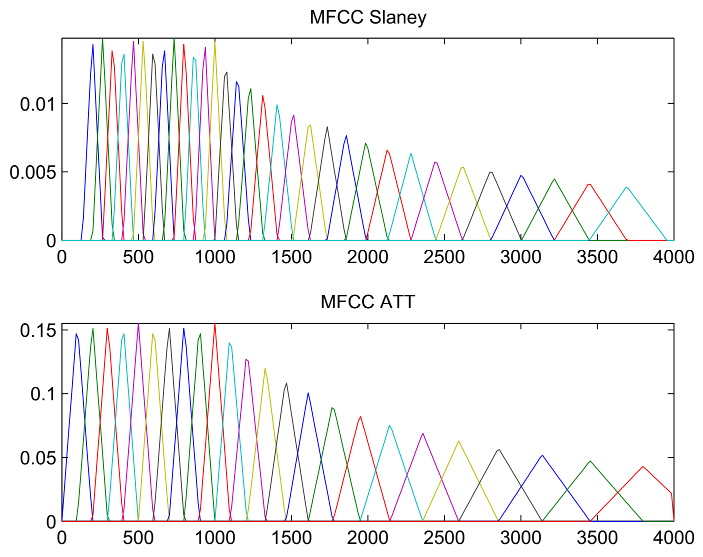


Figure 4.1: Top: The Mel Filter Bank consisting of 32 Equal Area triangular filters used in MFCC routine in Slaney's Auditory Toolbox; Bottom: The Mel Filter Bank consisting of 24 Equal Area triangular filters used in the AT&T MFCC routine

be fairly uncorrelated with the static MFCCs [9]. The difference coefficients on their own though, do not perform as well as static coefficients when used as features. This has been verified in the experiments in Chapter 8. In addition the experiments show how the performance of a speaker verification system is affected by using static features alone, static features concatenated with first order difference coefficients, and static features concatenated with first and second order difference coefficients.

4.2 A study of two feature extraction implementations

As a part of this work two different MFCC implementations, one from AT&T Labs, and the other from Malcom Slaney's Auditory Toolbox in MATLAB [31], were studied in detail. Details regarding these implementations are discussed in the following sections, while evaluation of performance of these implementations on the speaker verification task is presented in Chapter 8.

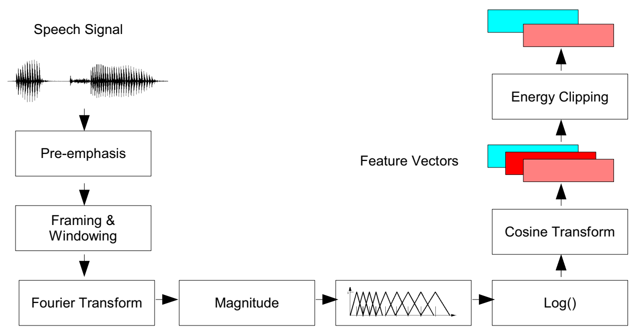


Figure 4.2: Steps involved in extracting MFCC feature vectors as implemented in Slaney's Auditory Toolbox in MATLAB

4.2.1 MFCC feature extraction in Malcom Slaney's Auditory Toolbox in MATLAB

An Overview of the Implementation

Figure 4.2 shows the steps involved in extracting MFCC feature vectors as implemented in Malcom Slaney's Auditory Toolbox in MATLAB.

The input speech is first pre-emphasized with a Finite Impulse Response(FIR) pre-emphasis filter with coefficients $[1\ -.97]$. The pre-emphasis is done in order to correct the spectral tilt in the speech spectrum. After pre-emphasis the feature extraction process is carried out by analyzing the speech signal in frames. A speech frame of size 20ms is extracted and multiplied with a Hamming window, to give a vector which we denote by $\mathbf{y}$. For speech sampled at 8Khz the window size is 160 samples, and the frame shift is 10ms or 80 samples. The Slaney implementation uses a 161 point Hamming window truncated to a 160 point Hamming window, at a sampling rate of 8Khz. A 512-point Discrete Fourier Transform $\{Y(k); k = 1, \ldots, 512\}$, of $\mathbf{y}$, using the native MATLAB FFT routine is then calculated. Subsequently, the absolute magnitude $\{|Y(k)|; k = 1, \ldots, 512\}$ and squared magnitude $\{|Y(k)|^2; k = 1, \ldots, 512\}$ are calculated. The squared magnitude, $\{|Y(k)|^2; k = 1, \ldots, 512\}$, is used to calculate the energy in the frame. The absolute magnitude signal denoted by, $\{|Y(k)|; k = 1, \ldots, 512\}$, is then passed through a Mel Filter Bank consisting of 32 triangular, equal area filters covering a bandwidth of 133.33Hz-3954.26Hz, to extract the energy in each frequency band covered by each of the 32 filters.

The filter bank used in this implementation is displayed in the top figure of Fig. 4.1. The center frequencies of the filters are warped according to the Mel scale, with the center frequencies being linear until 1000Hz, and logarithmic above 1000Hz. For $M = 32$ filters, M+2 boundary points are required to describe the filter bank(M center frequencies and 2 extreme boundary frequencies). Let $\{f_{c_i}\ \ i = 1, \ldots, M\}$ denote the center frequency of the ith filter. Also, let $\{f_{b_j}$ j = 1,. . . ,M+2} denote the $M + 2$ boundary points. It is important to note that for this implementation $\{f_{c_i} = f_{b_j}$ for $i = 1, \ldots, M$, and $j = 2, \ldots, M + 1\}$. To simplify notation, therefore, the boundary points for the ith filter could simply be stated by the points $f_{b_{i-1}}$ and $f_{b_{i+1}}$.

The center frequencies of the first 13 filters are linearly spaced, in the range [200-1000]Hz, with a spacing of 66.67Hz between the center frequency of each filter. The center frequencies of the remaining 19 filters are logarithmically spaced in the range [1071-3954.26]Hz. The center frequencies for these 19 filters are calculated as,

$$f_{c_i} = 1.0711703 \times f_{c_{i-1}} \quad i = 14, \ldots, M$$

The filter weights for the ith triangular filter,denoted H_i by are calculated as,

$$\begin{aligned} H_i(k) &= 0 & &\text{if } w(k) < f_{b_{i-1}} \\ &= \frac{2(w(k) - f_{b_{i-1}})}{(f_{c_i} - f_{b_{i-1}})(f_{b_{i+1}} - f_{b_{i-1}})} & &\text{if } f_{b_{i-1}} \le w(k) \le f_{b_i} \\ &= \frac{2(f_{b_{i+1}} - w(k))}{(f_{c_i} - f_{b_{i-1}})(f_{b_{i+1}} - f_{b_{i-1}})} & &\text{if } f_{b_i} \le w(k) \le f_{b_{i+1}} \\ &= 0 & &\text{if } w(k) > f_{b_{i+1}} \end{aligned}$$

where $\{k = 1, \ldots, 512\}$ denotes a DFT frequency index, and $\{w(k),\ k = 1, \ldots, 512\}$, is used to denote the physical DFT frequencies on the unit circle. The log energy output of the ith filter, denoted by mfe(i), is then calculated as,

$$\text{mfe}(i) = \log\,(\sum_{k=f_{b_{i-1}}}^{f_{b_{i+1}}} H_i(k)|Y(k)|). \tag{4.5}$$

Finally the cepstral coefficients are computed using an inverse Fourier transform, which for real signals can be implemented using a cosine transform. The mth cepstral coefficient is denoted by mfcc(m), where $\{m = 0, \ldots, C-1\}$ and C is the number of required

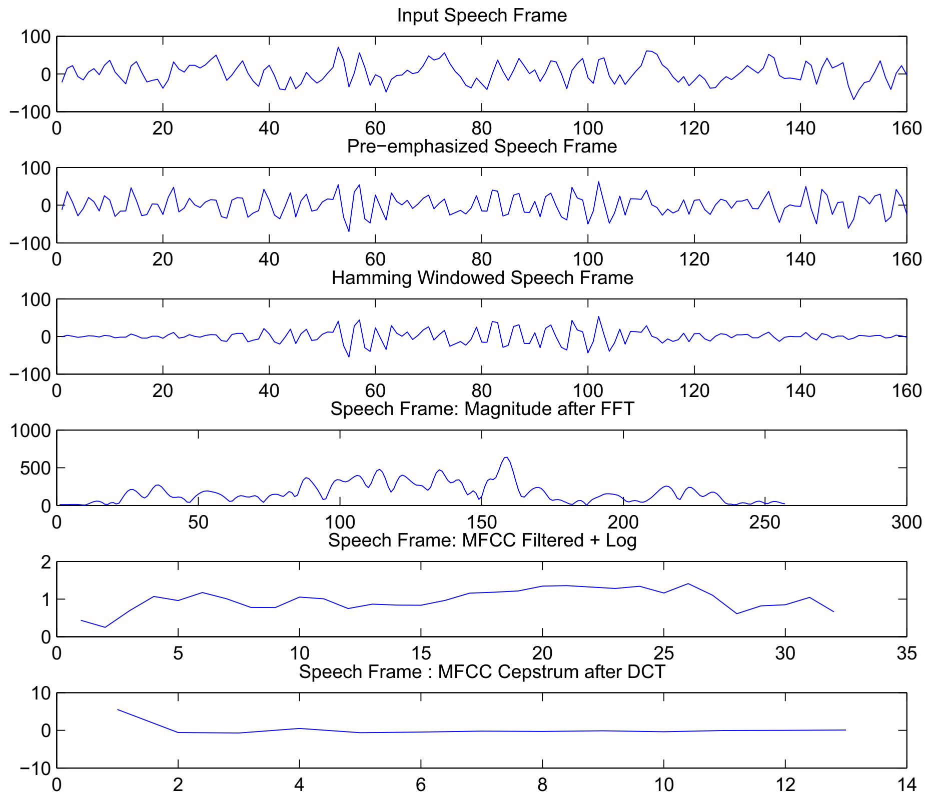


Figure 4.3: Graphs detailing outputs at each stage of the processing of the MFCC feature extraction program from the Auditory ToolBox

coefficients, is then calculated as,

$$\text{mfcc}(m) = \frac{1}{M}\sum_{l=1}^{M} \text{mfe}(l)\cos(m(l-\frac{1}{2})\frac{\pi}{M}) \quad m = 0,\ldots,C-1 \tag{4.6}$$

In the evaluations studied in this thesis we used $C = 13$ coefficients. The mfcc(0) component is the average log energy and is not used in the experiments in this thesis. Thus an extracted MFCC feature vector has a dimensionality of $D = 12$.

It is very informative to view the output at each stage during the processing of a speech frame. Fig. 4.3 displays the output of each stage for one such speech frame. The last of the six plots displays the value mfcc(0) coefficient due to which a vector of dimensionality $C = 13$ is seen instead of $D = 12$.

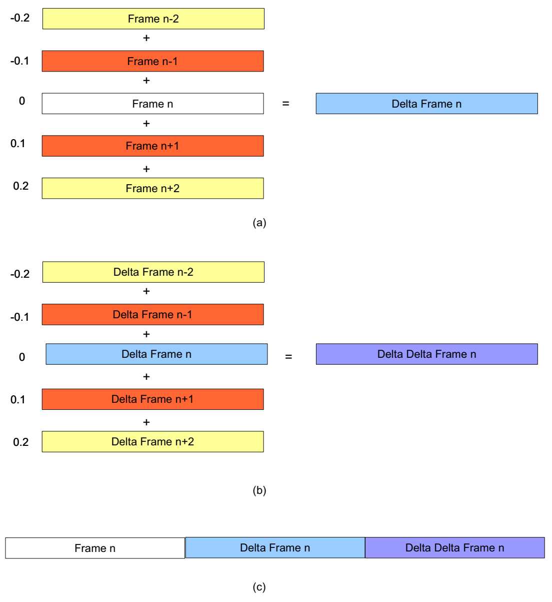


Figure 4.4: Delta and Delta-Delta Computation applied to feature vectors from Malcom Slaney's Auditory Toolbox; Fig(a)Delta Computation;Fig.(b)Delta-Delta Computation;Fig.(c)Concatenated feature vector for frame n

Calculation of the Delta Cepstrum

Fig. 4.4 shows how the calculations for the delta cepstrum and delta-delta cepstrum are implemented. As shown in Fig. 4.4(a) the delta cepstrum is calculated over ±2 speech frames, with corresponding frames being multiplied by the coefficients [-.2 -.1 0 .1 .2], and then added together. The delta-delta cepstrum values are similarly calculated over ±2 delta-cepstrum frames as shown in Fig. 4.4(b). The delta-cepstrum and delta-delta cepstrum are then simply concatenated with the concerned frame as shown in Fig. 4.4(c) to either give a feature vector of dimensionality $2D = 24$ or $3D = 36$, as required.

4.2.2 MFCC feature extraction implementation from AT&T Labs

An Overview of the implementation

Figure 4.5 shows the steps involved in extracting MFCC feature vectors as in the implementation from AT&T Labs.

A speech frame of size 20ms is extracted and multiplied by a Hamming window, to give a vector which we denote by $\mathbf{y}$. As mentioned in Section 4.2.1, for speech sampled

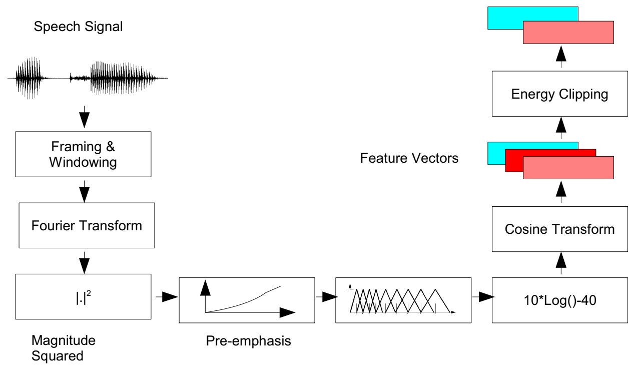


Figure 4.5: Steps involved in extracting MFCC feature vectors in the AT&T Labs implementation

at 8Khz the window size is 160 samples, and the frame shift is 10ms or 80 samples. A Hamming window of 160 samples is used in this implementation. Next a 512-point Discrete Fourier Transform $\{Y(k); k = 1, \ldots, 512\}$, of $\mathbf{y}$, is calculated using an FFT routine. The DFT coefficients computed by this routine and the native MATLAB FFT routine differ by a constant scale factor. Subsequently, the squared magnitude $\{|Y(k)|^2; k = 1, \ldots, 512\}$ is calculated, and is used to calculate the energy in the frame. The squared magnitude signal, $\{|Y(k)|^2; k = 1, \ldots, 512\}$, is then passed through a pre-emphasis filter thereby correcting the spectral tilt in the speech spectrum, to give the pre-emphasized signal which we denote by $\{A(k); k = 1, \ldots, 512\}$. If we allow $\{w(k),\ k = 1, \ldots, 512\}$, to denote the physical DFT frequencies on the unit circle, then coefficients for the pre-emphasis filter are calculated as,

$$P(k) = 1 + \frac{w(k)^2}{2.5 \times 10^5} \; k = 1, \ldots, 512. \tag{4.7}$$

The pre-emphasized signal $\{A(k); k = 1, \ldots, 512\}$ is computed as,

$$A(k) = |Y(k)|^2 P(k) \; ; k = 1, \ldots, 512. \tag{4.8}$$

The pre-emphasized signal $\{A(k); k = 1, \ldots, 512\}$ is then passed through a Mel Filter Bank consisting of 24 triangular, equal area filters covering a bandwidth of 0Hz-4000Hz, to extract the energy in each each frequency band covered by each of the 24 filters.

The filter bank used in this implementation is shown in the bottom figure of Fig. 4.1. Similar to the filter bank described in Section 4.2.1, the center frequencies of the filters are warped according to the Mel scale, with the center frequencies being linear till

1000Hz, and logarithmic above 1000Hz. Notation developed in Section 4.2.1 is used to describe the parameters of the Mel Filter Bank used in this implementation and is repeated here for convenience. $M = 24$ is the number of triangular filters, $\{f_{c_i}$ i=1,...,M$\}$ to denote the center frequencies of the triangular filters, and the boundary points for the ith filter are the points $f_{b_{i-1}}$ and $f_{b_{i+1}}$ where $\{f_{b_j}$ j = 1,...,M+2$\}$ are the $M + 2$ boundary points and $\{f_{c_i} = f_{b_j}$ for $i = 1, \ldots, M$, and $j = 2, \ldots, M + 1\}$.

The center frequencies of the first 10 filters are linearly spaced, in the range [100-1000]Hz, with a spacing of 100.00Hz between the center frequency of each filter. The center frequencies of the 14 remaining filters are logarithmically spaced in the range [1100-3797.5]Hz. The center frequencies for the logarithmically spaced filters are calculated as,

$$f_{c_i} = 1.1 \times f_{c_{i-1}} \quad i = 11, \ldots, M$$

The filter weights for the ith triangular filter,denoted T_i by are calculated as,

$$\begin{aligned} T_i(k) &= 0 && \text{if } w(k) < f_{b_{i-1}} \\ &= \frac{(w(k) - f_{b_{i-1}})}{(f_{c_i} - f_{b_{i-1}})} && \text{if } f_{b_{i-1}} \le w(k) \le f_{b_i} \\ &= \frac{(f_{b_{i+1}} - w(k))}{(f_{c_i} - f_{b_{i-1}})} && \text{if } f_{b_i} \le w(k) \le f_{b_{i+1}} \\ &= 0 && \text{if } w(k) > f_{b_{i+1}} \end{aligned}$$

This equation gives a Mel Filterbank identical to the one mentioned by Davis et al. in [6]. Each filter T_i is then normalized to give equal area filters denoted by H_i,

$$H_i(k) = \frac{T_i(k)}{\sum_{w(k)=f_{b_{i-1}}}^{w(k)=f_{b_{i+1}}} T_i(k)}$$

where $\{k = 1, \ldots, 512\}$ denotes a DFT frequency index, and $\{w(k),\ k = 1, \ldots, 512\}$, is used to denote the physical DFT frequencies on the unit circle. As seen in Fig. 4.1, the last filter is clipped off just before 4000Hz, to prevent aliasing. The log energy output of the ith filter, denoted by mfe(i), is then calculated as,

$$\text{mfe}(i) = 10 \log \Big(\sum_{w(k)=f_{b_{i-1}}}^{f_{b_{i+1}}} H_i(k) A(k) \Big) - 40. \tag{4.9}$$

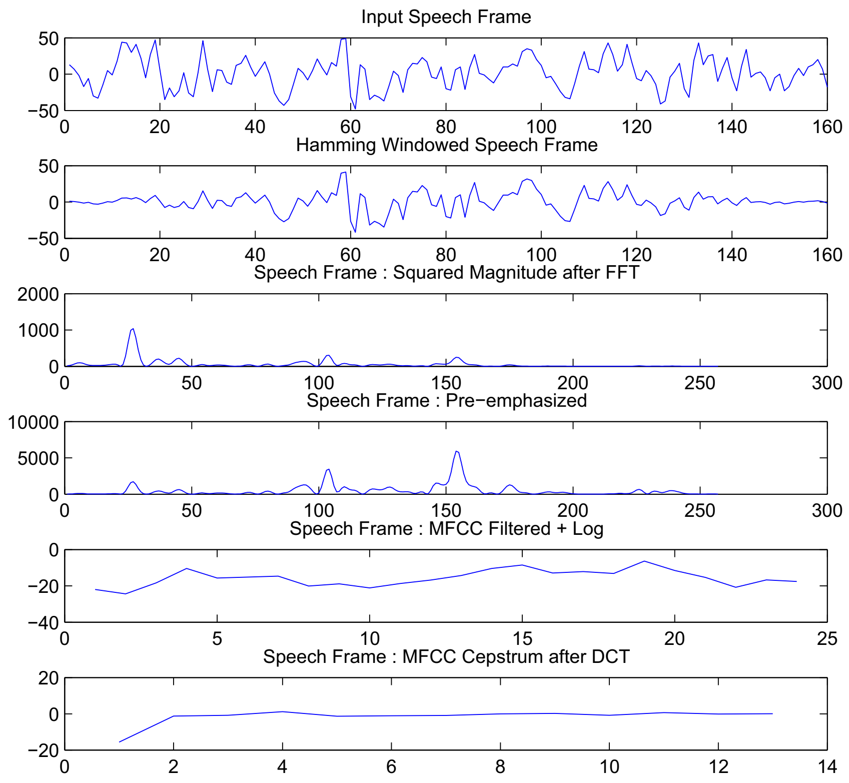


Figure 4.6: Graphs detailing outputs at each stage of the processing of the MFCC feature extraction program from AT&T Labs

A discrete cosine transform as is then applied to the log energy calculated in Eq. (4.9) to obtain the Mel Frequency Cepstral Coefficients denoted by mfcc. The coefficient mfcc(0), which represents the average log energy, is discarded, and the resulting dimensionality of the vector is $D = 12$.

It is very informative to view the output at each stage during the processing of a speech frame. Fig. 4.6 displays the output of each stage for one such speech frame. The last of the six plots displays the value mfcc(0) coefficient due to which a vector of dimensionality $C = 13$ is seen instead of $D = 12$.

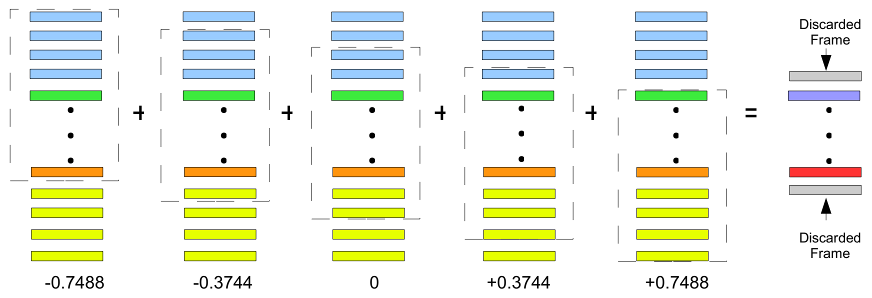


Figure 4.7: Computation of Delta Coefficients in the AT&T MFCC feature extraction program

Calculation of the Delta Cepstrum

The technique for calculation of the delta cepstrum in this implementation is shown in Fig. 4.7. We assume that MFCC feature extraction, as explained in Section 4.2.2, yields a stack of N feature vectors or frames. Three copies of the first frame are appended to the head of the stack of N frames and similarly three copies of the last frame are appended to the tail of the stack of N frames. The delta cepstrum is calculated over five "sliding" windows each of size $N+2$ frames. The first window of extends from the first copy of the first frame, to the $N-1$th frame. At every stage, the "sliding" window advances by one frame. All the frames within the range of the window are multiplied by the corresponding coefficient in the vector $[-0.7488\ \ -0.3744\ 0\ \ +0.3744\ \ +0.7488]$, at a given stage of a calculation as shown in Fig. 4.7. The results from each stage are then added up to yield a stack of $N+2$ frames, in which the first and last frames are redundant, and are discarded. The N frames that are thus retained are the required delta cepstrum frames. A similar procedure is used for the calculation of the second difference MFCC coefficients.

4.3 Energy Computation and Energy Clipping

It was mentioned in Sections 4.2.1 and 4.2.2 that the squared magnitude of the Discrete Fourier Transform $\{|Y(k)|^2; k = 1, \ldots, 512\}$, of the windowed portion $\mathbf{y}_n$ of a speech utterance is used in the computation of the energy in the frame. The energy in the speech frame $\mathbf{y}_n$ is thus calculated using Parseval's Equation, and is given by

$$E(\mathbf{y}_n) = \frac{1}{N_{FFT}} \sum_{k=0}^{N_{FFT}-1} |Y(k)|^2 \tag{4.10}$$

where E denotes the short-time energy of a speech vector $\mathbf{y}_n$ and $N_{FFT} = 512$ denotes the number of points in the FFT. For each frame the log energy is then computed as,

$$E_{log}(\mathbf{y}_n) = 10 \log_{10}(E(\mathbf{y}_n)) - 40.$$

Once the log energy for each frame has been computed, the energy in dB, E_{dB}, is then computed for each frame, relative to the frame in the utterance which has the maximum log energy. If E_{log}^{max} is used to denote the maximum log energy value encountered in a given set of frames, then the energy in dB for a frame is computed as,

$$E_{dB}(\mathbf{y}_n) = E_{log}(\mathbf{y}_n) - E_{log}^{max}$$

To perform energy clipping once an energy threshold value E_{thr} in dB is specified, any frame whose energy satisfies the relation $E_{dB} < E_{thr}$ is simply dropped to give a set of feature vectors whose energy is above the specified energy threshold.

Chapter 5

Pattern Recognition for Speaker Verification

5.1 The Gaussian Mixture Model

A Gaussian Mixture density can be written as a linear superposition of K Gaussian densities of the form:

$$p(\mathbf{x}|\mathbf{\Lambda}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_k, \mathbf{\Sigma}_k) \tag{5.1}$$

where $\mathbf{x}$ is a D-dimensional MFCC speech vector, $\mathcal{N}(\mathbf{x}|\mu_k, \mathbf{\Sigma}_k)$, $k = 1, \ldots, K$, are the component Gaussian densities and π_k, $k = 1, \ldots, K$, are called the *mixing coefficients or mixture weights*. Each component Gaussian density is a D-variate Gaussian of the form:

$$\mathcal{N}(\mathbf{x}|\mu_k, \mathbf{\Sigma}_k) \ = \frac{1}{(2\pi)^{D/2}|\mathbf{\Sigma}_k|^{1/2}} \exp\{-\frac{1}{2}(\mathbf{x} - \mu_k)'\mathbf{\Sigma}_k^{-1}(\mathbf{x} - \mu_k)\} \tag{5.2}$$

with a $D \times 1$ mean vector μ_k and a $D \times D$ covariance matrix $\mathbf{\Sigma}_k$. All mixing coefficients satisfy the conditions:

$$\sum_{k=1}^{K} \pi_k = 1. \tag{5.3}$$

and

$$0 \leq \pi_k \leq 1 \tag{5.4}$$

which ensures the mixture is a true probability density function. A Gaussian Mixture density can thus be parameterized by:

$$\mathbf{\Lambda} \equiv [\mu_k, \mathbf{\Sigma}_k, \pi_k] \qquad k = 1, \ldots, K. \tag{5.5}$$

which is used to denote the set of mixture wights, mean vectors and covariance matrices for each of the K component mixture densities.

5.2 Reasons for using GMMs in Text-Independent Speaker Recognition[1]

The first main motivation for using GMMs is the intuitive notion that the component densities of a multi-variate density, such as a mixture of Gaussians, model a set of underlying acoustic classes. It can be assumed that the acoustic space corresponding to a speaker's voice can be characterized by a set of acoustic classes representing some broad phonetic events such as vowels, nasals, or fricatives. These acoustic classes reflect some general speaker-dependant vocal tract configurations that are useful for characterizing speaker identity. The spectral shape of the kth acoustic class can be represented by the mean μ_k and the variations of the average spectral shape can be represented by the covariance matrix $\mathbf{\Sigma}_k$. Since the training and testing of speech is unlabeled or unsupervised, the acoustic classes are "hidden" in that the class of an observation is unknown. With the assumption of independent feature vectors, the observation density of feature vectors drawn from these hidden acoustic classes is thus a Gaussian mixture.

Another motivation for using GMMs is that a linear superposition of a set of Gaussian basis functions is capable of representing a large class of sample distributions. One of the powerful attributes of the GMM is its ability to form smooth approximations to arbitrarily shaped densities. The classical unimodal Gaussian speaker model represents a speaker's feature distribution by a position(mean vector) and an elliptic shape(covariance). A VQ code-book, on the other hand represents a speaker's distribution by a discrete set of characteristic templates. In a K component VQ code-book the kth vector could be thought of as a template that is representative of a feature from a particular acoustic class. In the acoustic feature space these could be thought of as a set of impulses at the points whose co-ordinates are represented by the individual elements of the kth vector. To allow some variability in the representation of each of the K acoustic classes the K impulses could instead be replaced by K multi-variate Gaussians with respective means and covariances, this giving rise to a GMM. Thus a discrete set of Gaussian functions, each with their own mean and variance, model the

[1]**Material in this section is borrowed from Section II-C of [9]**

set of underlying acoustic classes better when compared to a discrete VQ code-book or a unimodal Gaussian.

Further, because the component Gaussians are acting together to model the overall pdf, full covariance matrices are not necessary even if the features are not statistically independent. The linear combination of diagonal covariance Gaussians is capable of modeling the correlations between feature vector elements. The effect of using a set of M full covariance Gaussians can be equally obtained by using a larger set of diagonal covariance Gaussians.

5.3 The Gaussian Mixture Model as a generative model

Given a set of observation feature vectors $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^{T}$, one can view the production of an observation vector $\mathbf{x}_n$ from a Gaussian Mixture density as a generative process. A "hidden" trial is first performed where the mixture component(or the acoustic class) k is picked with a prior probability π_k, following which $\mathbf{x}_n$ is picked from the kth Gaussian. Performing this "hidden" trial could be quantified by introducing a binary random variable $\mathbf{z}_n$, such that choosing the k^{th} component of the Gaussian mixture would mean that the kth element of $\mathbf{z}_n$, $z_{nk} = 1$ and $z_{nj} = 0$ for $j \neq k$, $\forall j, k = 1 \ldots\ K$ and $\sum_{k=1}^{K} z_{nk} = 1$. Therefore $\mathbf{z}_n$ could be in one of K possible states. With this interpretation, a complete description of the generative process is available when the set of feature vectors $\mathbf{X}$ is accompanied by a set of corresponding random variables $\mathbf{Z} = \{\mathbf{z}_n\}_{n=1}^{T}$ from which $\mathbf{z}_n$ would indicate the "hidden" trial that was performed in order to generate an observation vector $\mathbf{x}_n$. This generative process is illustrated in Figure 5.1

The probability of performing the hidden trial can be expressed in terms of the mixing coefficients as,

$$\begin{aligned} p(z_{nk} = 1) &= \pi_k \\ p(\mathbf{z}_n) &= \prod_{k=1}^{K} \pi_k^{z_{nk}} \end{aligned} \tag{5.6}$$

and the probability of drawing the vector $\mathbf{x}_n$ conditioned on performing this hidden trial can be expressed as,

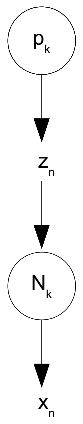


Figure 5.1: Illustration of the GMM as a generative model generating an observation sample $\mathbf{x_n}$.p_k is the prior probability π_k and N_k is a component Gaussian $\mathcal{N}(\mathbf{x}|\mu_k, \mathbf{\Sigma}_k)$

$$\begin{aligned} p(\mathbf{x}_n|z_{nk} = 1) &= \mathcal{N}(\mathbf{x}_n|\mu_k, \mathbf{\Sigma}_k) \qquad (5.7) \\ p(\mathbf{x}_n|\mathbf{z}_n) &= \prod_{k=1}^{K} \mathcal{N}(\mathbf{x}_n|\mu_k, \mathbf{\Sigma}_k)^{z_{nk}} \end{aligned}$$

As stated in [8], the generative process for drawing each of the observation vectors in the set $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^{T}$ is very similar to that of a HMM, except that for a GMM there are no Markovian constraints for the selection of classes since the text is unconstrained and it is unlikely that there is any speaker specific information in acoustic class sequencing. Therefore, GMM models only the acoustic classes which comprise a person's speech but not the temporal ordering of the acoustic classes.

5.4 Estimation

The process of "training" consists of obtaining estimates for the parameters that characterize a probability distribution. Both Maximum A Posteriori(MAP) and Maximum Likelihood(ML) estimation methods are used to obtain estimates of these parameters. Using notation from [32], if we let θ denote a random vector taking values in the space Θ as the parameter vector to be estimated from a set of observations $\mathbf{X}$ which are assumed to be drawn from a probability density function(pdf) $p(.|\theta)$, and if g is the prior pdf over θ then the MAP estimate, θ_{MAP}, is defined as the mode of the posterior pdf of θ denoted as $g(.|\mathbf{X})$,i.e.

$$\theta_{MAP} = \arg\max_{\theta} g(\theta|\mathbf{X}) \tag{5.8}$$

$$= \arg\max_{\theta} p(\mathbf{X}|\theta)g(\theta) \tag{5.9}$$

If the prior pdf g is non-informative or in other words if all values of θ in the parameter space Θ are equally likely then the MAP estimate θ_{MAP} is equivalent to the ML estimate denoted by θ_{ML} and is equivalently estimated by

$$\theta_{ML} = \arg\max_{\theta} p(\mathbf{X}|\theta) \tag{5.10}$$

Equivalently, the maximum likelihood estimate θ_{ML} is one that satisfies the equation,

$$\theta_{ML} = \frac{\partial p(\mathbf{X}|\theta)}{\partial \theta} = 0 \tag{5.11}$$

The quantity $p(\mathbf{X}|\theta)$ is by definition called the *likelihood function* or simply the *likelihood* of the p.d.f $p(.|\theta)$ given the data set $\mathbf{X}$ and Eq. (5.11) is called the likelihood equation.

The Gaussian Mixture density is characterized by the set of parameters mentioned in Eq. (5.5) namely, the mean vectors, mixture weights and the covariance matrices of the mixture components. This chapter discusses the MAP and ML estimation procedures to obtain estimates of these parameters.

5.5 ML Estimation and the EM Algorithm

In viewing the GMM as a generative model as in Section 5.3 it is evident that a *complete* description of the data is provided by specifying the observation feature vectors $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^{T}$ and by specifying the component distribution $\mathbf{Z} = \{\mathbf{z}_n\}_{n=1}^{T}$ from the Gaussian mixture density that generated each one of them. In unsupervised learning problems such as the estimation of parameters $\mathbf{\Lambda}$ of a Gaussian Mixture density the random variables in set $\mathbf{Z}$ are never really observed, and in fact remain *hidden* or *latent*. The only observable quantities are the observation vectors $\mathbf{X}$ from which the parameters for the Gaussian Mixture density need to be estimated. Therefore this is an estimation problem with *incomplete*, *latent*, or *hidden* data. Estimation of parameters of a distribution from incomplete data can be achieved by an iterative algorithm called the Expectation-Maximization(EM) Algorithm[33]. The basic idea of the EM Algorithm is, given an initial estimate of the model $\mathbf{\Lambda}$, to estimate a new model $\mathbf{\Lambda}^{'}$, such that $p(\mathbf{X}|\mathbf{\Lambda}^{'}) \geq p(\mathbf{X}|\mathbf{\Lambda})$. The new model then becomes the current model and the process is repeated until some convergence threshold is reached [8]. ML Estimation via the EM Algorithm was first discussed in [33] and specifically for Gaussian Mixtures is discussed in [8, 9, 34]. Though [33] mentions that the EM Algorithm is also applicable in the MAP estimation framework, a detailed treatment of MAP estimation of parameters for a GMM via the EM algorithm is provided in [4, 32]. The discussion regarding Bayesian Adaptation and MAP Estimation for GMMs is deferred until Section 5.7

5.5.1 The Need for an Auxiliary Function

For the sake of completeness it is necessary to mention the complete data likelihood $p(\mathbf{X},\mathbf{Z}|\mathbf{\Lambda})$ and how the incomplete data likelihood $p(\mathbf{X}|\mathbf{\Lambda})$ develops from this quantity.

Given $\mathbf{X}$ and $\mathbf{Z}$, from Eqs. (5.6) and (5.7) the complete data likelihood $p(\mathbf{X},\mathbf{Z}|\mathbf{\Lambda})$, is given by,

$$p(\mathbf{X},\mathbf{Z}|\mathbf{\Lambda}) = \prod_{n=1}^{T}\prod_{k=1}^{K}\pi_k^{z_{nk}}[\mathcal{N}(\mathbf{x}_n|\mu_k,\mathbf{\Sigma}_k)]^{z_{nk}} \tag{5.12}$$

where we let z_{nk} to be the k^{th} component of $\mathbf{z}_n$. Then the observed(incomplete) data likelihood $p(\mathbf{X}|\mathbf{\Lambda})$ can be written as,

$$\begin{aligned} p(\mathbf{X}|\mathbf{\Lambda}) &= \sum_{Z} p(\mathbf{X},\mathbf{Z}|\mathbf{\Lambda}) && (5.13)\\ &= \prod_{n=1}^{T}\sum_{k=1}^{K}\pi_k\mathcal{N}(\mathbf{x}_n|\mu_k,\mathbf{\Sigma}_k) && (5.14)\end{aligned}$$

by marginalizing the complete data likelihood $p(\mathbf{X},\mathbf{Z}|\mathbf{\Lambda})$ over all possible acoustic states $\mathbf{Z}$.

It can be shown that, given a model $\mathbf{\Lambda}$ and the data $\mathbf{X}$, we can, in reality, obtain only posterior knowledge of the latent variables $\mathbf{Z}$ by computing the quantity $p(\mathbf{Z}|\mathbf{X},\mathbf{\Lambda})$ from Eqs. (5.6) and (5.7) and the Bayes' Theorem as,

$$p(\mathbf{Z}|\mathbf{\Lambda}, \mathbf{X}) \propto \prod_{n=1}^{T}\prod_{k=1}^{K}\pi_k^{z_{nk}}\mathcal{N}(\mathbf{x}_n|\mu_k, \mathbf{\Sigma}_k)^{z_{nk}} \tag{5.15}$$

In estimation, for analytical(and computational) purposes, it is usually easier to work with the logarithm of the likelihood rather than the likelihood itself [20]. In problems of parameter estimation of probability distributions involving latent variables, it is usually much more tractable to work with the joint log-likelihood over the complete data i.e., $\ln p(\mathbf{X}, \mathbf{Z}|\mathbf{\Lambda})$ rather than the log-likelihood of the incomplete data $p(\mathbf{X}|\mathbf{\Lambda})$. Baum et al. in [35] suggested the use of an auxiliary function $Q(\mathbf{\Lambda}, \mathbf{\Lambda}^{'})$ that makes use of $\ln p(\mathbf{X}, \mathbf{Z}|\mathbf{\Lambda})$ and allows achieving the objective of estimating a new model $\mathbf{\Lambda}^{'}$, such that $p(\mathbf{X}|\mathbf{\Lambda}^{'}) \geq p(\mathbf{X}|\mathbf{\Lambda})$, where $\mathbf{\Lambda}$ is a previous estimate of the model.

5.5.2 Maximum Likelihood and The EM Algorithm

For ML estimation auxiliary function $Q(\mathbf{\Lambda}, \mathbf{\Lambda}^{'})$ is,

$$Q(\mathbf{\Lambda}, \mathbf{\Lambda}^{'}) = \sum_{Z} \ln\left(p(\mathbf{X}, \mathbf{Z}|\mathbf{\Lambda}^{'})\right) p(\mathbf{Z}|\mathbf{\Lambda}, \mathbf{X}) \tag{5.16}$$

The expression for the auxiliary function in Eq. (5.16) is also viewed as the conditional expected value of the complete data log-likelihood $\ln\left(p(\mathbf{X}, \mathbf{Z}|\mathbf{\Lambda}^{'})\right)$,taken with respect to the conditional distribution of $\mathbf{Z}$, $p(\mathbf{Z}|\mathbf{\Lambda}, \mathbf{X})$, given $\mathbf{\Lambda}$ and the set of observations $\mathbf{X}$.

An analytical expression for the auxiliary function, as stated in Eq. (5.16), can be obtained by making use of Eqs. (5.12) and (5.15) as,

$$Q(\mathbf{\Lambda}, \mathbf{\Lambda}^{'}) = \sum_{n=1}^{T}\sum_{k=1}^{K}\gamma(z_{nk})\{\ln \pi_k^{'} + \ln \mathcal{N}(\mathbf{x}_n|\mu_k^{'}, \mathbf{\Sigma}_k^{'})\} \tag{5.17}$$

where $\gamma(z_{nk})$ is given by,

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \mathbf{\Sigma}_k)}{\sum_{l=1}^{K}\pi_l \mathcal{N}(\mathbf{x}_n|\mu_l, \mathbf{\Sigma}_l)} \quad n = 1, 2, \ldots, T \tag{5.18}$$

The ML parameter estimates, for the new model $\mathbf{\Lambda}^{'} = [\mu_k^{'}, \mathbf{\Sigma}_k^{'}, \pi_k^{'}]$ $k = 1, \ldots, K$ can be obtained by setting $\frac{\partial Q(\mathbf{\Lambda}, \mathbf{\Lambda}^{'})}{\partial \mathbf{\Lambda}^{'}} = 0$ instead of setting the partial derivative of the likelihood function, as in Eq. (5.11), with respect to the parameters to zero.

The ML Estimate of the mean $\mu_k^{'}$ is obtained by setting $\frac{\partial Q(\mathbf{\Lambda}, \mathbf{\Lambda}')}{\partial \mu_k^{'}} = 0$ and is given by,

$$\mu_k^{'} = \frac{1}{T_k} \sum_{n=1}^{T} \gamma(z_{nk}) \mathbf{x}_n \tag{5.19}$$

The ML estimate of the covariance matrix $\mathbf{\Sigma}_k^{'}$ is given by setting $\frac{\partial Q(\mathbf{\Lambda}, \mathbf{\Lambda}^{'})}{\partial \mathbf{\Sigma}_k^{'}} = 0$ and is given by,

$$\mathbf{\Sigma}_k^{'} = [\frac{1}{T_k} \sum_{n=1}^{T} \gamma(z_{nk}) \mathbf{x}_n^2] - \mu_k^{'2} \tag{5.20}$$

The ML estimate of the mixture weights $\pi_k^{'}$ is given by setting $\frac{\partial Q(\mathbf{\Lambda}, \mathbf{\Lambda}^{'})}{\partial \pi_k^{'}} = 0$ and is given by,

$$\pi_k^{'} = \frac{T_k}{T} \tag{5.21}$$

where

$$T_k = \sum_{n=1}^{T} \gamma(z_{nk}) \tag{5.22}$$

The EM Algorithm for ML Estimation can thus be summarized as:

Given a model $\mathbf{\Lambda}$ for a Gaussian Mixture Model with parameters $[\mu_k, \mathbf{\Sigma}_k, \pi_k]$ for $k = 1, \ldots, K$ we obtain a new model $\mathbf{\Lambda}^{'}$ with parameters $[\mu_k^{'}, \mathbf{\Sigma}_k^{'}, \pi_k^{'}]$ for $k = 1, \ldots, K$ such that $p(\mathbf{X}|\mathbf{\Lambda}^{'}) \geq p(\mathbf{X}|\mathbf{\Lambda})$,

E Step Using the old model $\mathbf{\Lambda}$ compute the quantity $\gamma(z_{nk})$ as in Eq. (5.18).

M Step After computing $\gamma(z_{nk})$ obtain the parameter estimates of the new model $\mathbf{\Lambda}^{'}$ using Eqs. (5.19), (5.20), (5.21) and (5.22)

Update Compute the log-likelihood as $\log p(\mathbf{X}|\mathbf{\Lambda}^{'})$ and check for the convergence of the log-likelihood. If the convergence criterion is not satisfied then update $\mathbf{\Lambda} \leftarrow \mathbf{\Lambda}^{'}$ and return to the E Step.

5.5.3 Initializing the EM

The initial estimate to the EM algorithm for $\mathbf{\Lambda}$, is obtained by constructing a VQ code-book using the Linde-Buzo-Gray(LBG) Algorithm [19]. The following algorithm was taken directly from [19].

Before stating the algorithm it is necessary to mention the distortion measure used to calculate the distance between two vectors. The current implementation of the LBG Algorithm uses the Euclidean distance between two vectors as the distortion measure. Given two vectors of dimensionality D, $\mathbf{p} = [p_1, \ldots, p_D]$ and $\mathbf{q} = [q_1, \ldots, q_D]$,

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{D} (p_i - q_i)^2} \tag{5.23}$$

(0) Initialization: Fix $K = 2^R$, where R is an integer, and K is the largest number of levels(codebook size) desired. Fix $N = 10$(the maximum number of iterations), $T =$ length of training sequence, $\epsilon = 0.005$. Let M denote the intermediate number of levels in the codebook (which doubles after each binary split), until $M = K$, the final number of required levels is reached. Initially, set $M = 1$.

Given a training sequence $\mathbf{X} = [\mathbf{x_1}, \ldots, \mathbf{x_T}]$ set $A = \mathbf{X}$, the training sequence alphabet. Define $\hat{A}(1) = \hat{\mathbf{x}}(A)$, the centroid of the entire training sequence by setting it as the mean of all the training vectors.

(1) (Splitting) Given $\hat{A}(M) = \{y_i,\ i = 1, \ldots, M\}$ split each reproduction(codebook) vector y_i into $y_i + \epsilon$ and $y_i - \epsilon$, where ϵ is a fixed perturbation vector. Set $\hat{A}_0(2M) = \{y_i + \epsilon, y_i - \epsilon\ \ i = 1, \ldots, M\}$ and then replace M by $2M$.

(2) Let m denote the mth iteration, until convergence or until the maximum number of iterations $N = 10$ is reached, for an intermediate code-book with M-levels. Initially, set $m = 0$ and $D_{-1} = \infty$.

(3) Given $\hat{A}_m(M) = \{y_1, \ldots, y_M\}$,find its optimum partition $\mathcal{P}(\hat{A}_m(M)) = \{S_i,\ i = 1, \ldots, M\}$ that is, $\mathbf{x_j} \epsilon S_i$ if $d(\mathbf{x_j}, y_i) \leq d(\mathbf{x_j}, y_l)$, all l. Compute the resulting distortion

$$\begin{aligned} D_m &= D(\{\hat{A}_m(M), \mathcal{P}(\hat{A}_m(M))\}) \\ &= T^{-1}\sum_{j=1}^{T} \min_{y \epsilon \hat{A}_m} d(\mathbf{x_j}, y) \end{aligned}$$

(4) (Convergence) If $(D_{m-1} - D_m)/D_m \leq \epsilon$, then go to step (6). Otherwise continue.

(5) Find optimal reproduction alphabet $\hat{A}_{m+1}(M) = \hat{\mathbf{x}}(\mathcal{P}(\hat{A}_m(M))) = \{\hat{\mathbf{x}}(S_i);\ i = 1, \ldots, M\}$ for $\mathcal{P}(\hat{A}_m(M))$. Replace m by $m+1$ and go to (3).

(6) Set $\hat{A}(M) = \hat{A}_m(M)$. The final M-level quantizer is described by $\hat{A}(M)$. If $M = K$, halt with final quantizer described by $\hat{A}(K)$, otherwise go back to step (1).

The final codebook is then,

$$\begin{aligned} \hat{A}(K) &= \hat{\mathbf{x}}(\mathcal{P}(\hat{A}(K))) \\ &= \hat{\mathbf{x}}(S_i);\ i = 1, \ldots, K \\ &= \mu_k;\ k = 1, \ldots, K \end{aligned}$$

The number of vectors η_k in each partition $\{(S_k);\ k = 1, \ldots, K\}$, is preserved, and divided by the total number of vectors T, to give the set of mixture weights as,

$$\pi_k = \eta_k / T;\ k = 1, \ldots, K$$

And the covariance matrices $\Sigma_k;\ k = 1, \ldots, K$ are calculated as,

$$\mathbf{\Sigma}_k = [\frac{1}{\eta_k} \sum_{\mathbf{x_j} \epsilon S_k} \mathbf{x}_j^2] - \mu_k^2;\ k = 1, \ldots, K$$

5.6 Speaker Verification as a Hypothesis Testing Problem

Hypothesis testing is a process of establishing the validity of a hypothesis [36]. In the verification task an identity claimant I claims to be a speaker S and provides a voice sample $\mathbf{X}$ to the system. The system then needs to determine if I and S are the same person from an observation $\mathbf{X}$ provided by I. H_0 would then be the hypothesis that $\mathbf{X}$

is from the speaker S. This is called the *null* hypothesis. A complementary hypothesis H_1 would be the that $\mathbf{X}$ is not from the speaker S. This hypothesis is also called the *alternative* hypothesis.

Mathematically, H_0 is represented by a model $\mathbf{\Lambda}$ which denotes the parameters of the Gaussian Mixture density that characterizes the customer's voice. We call $\mathbf{\Lambda}$ the *speaker* model. Similarly H_1 is mathematically represented by $\bar{\mathbf{\Lambda}}$ which denotes the parameters of a Gaussian Mixture density that characterizes the set of impostors. $\bar{\mathbf{\Lambda}}$ is referred to as the *anti-speaker* or *background* model.

Given these two probability density functions, two *likelihoods* namely the likelihood of the speech utterance $\mathbf{X}$ given $\mathbf{\Lambda}$, denoted by $p(\mathbf{X}|\mathbf{\Lambda})$, and the likelihood of $\mathbf{X}$ given $\bar{\mathbf{\Lambda}}$, denoted by $p(\mathbf{X}|\bar{\mathbf{\Lambda}})$. The ratio of these two likelihoods, called the likelihood-ratio statistic, is computed and compared against a threshold θ, which depends on prior probabilities, to either accept the null hypothesis H_0 or accept the alternative hypothesis H_1. More formally, this verification decision test can be summed up as,

$$\frac{p(\mathbf{X}/\mathbf{\Lambda})}{p(\mathbf{X}/\bar{\mathbf{\Lambda}})} \begin{cases} \geq \theta & \text{accept } H_0 \\ < \theta & \text{accept } H_1 \end{cases} \tag{5.24}$$

which is the equation for a likelihood-ratio detector.

The evaluations used in this thesis make use of the ratio of the average log-likelihood score $L(\mathbf{X})$ given by,

$$L(\mathbf{X}) = \frac{1}{T}\log p(\mathbf{X}|\mathbf{\Lambda}) - \frac{1}{T}\log p(\mathbf{X}|\bar{\mathbf{\Lambda}}) \tag{5.25}$$

where T denotes the total number of feature vectors in the set $\mathbf{X}$, and the likelihoods $p(\mathbf{X}/\mathbf{\Lambda})$ and $p(\mathbf{X}/\bar{\mathbf{\Lambda}})$ are calculated as mentioned in Eq. (5.14) given the models $\mathbf{\Lambda}$ and $\bar{\mathbf{\Lambda}}$.

System evaluation consists of scoring a set of utterances from the valid speaker(or customer or client) and a set of utterances from designated impostors, to obtain two sets of log-likelihood ratio scores. The figure on the left hand side of Figure 5.2, displays histogram plots of the log-likelihood scores from both of these classes, namely the set of *client scores* or Class I and the set of *imposter scores* or Class II. Picking a

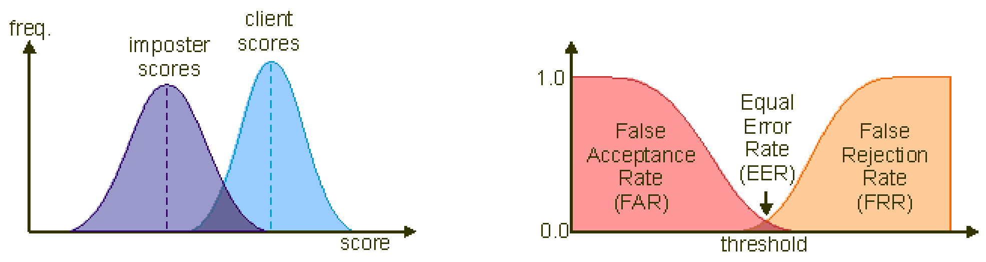


Figure 5.2: Illustration of the EER

threshold, θ, as specified in Eq. (5.24) involves choosing a point on the *score* axis of this histogram plot, to decide a boundary between the two classes. Once the threshold is chosen there are instances from both classes that would be misclassified as being from the other, based on the scores that are observed. The case where an imposter score is misclassified as a customer score is called a *false acceptance*, and the case where a customer score is misclassified as an impostor score is called a *false rejection*. The frequency of instances being misclassified on either side, accordingly gives us the probability of false acceptances(FA), denoted by $p(FA)$ or the False Acceptance Rate(FAR) and the probability of false rejections(FR), denoted by $p(FR)$ or the False Rejection Rate(FRR). The $p(FA)$ is the area under the histogram(or the probability density function approximation) of Class II that is misclassified as being from Class I and vice versa with $p(FR)$. Choosing a threshold θ then, is always a tradeoff between acceptable rates of FA and FR, depending on the task at hand. There exists a threshold where the FAR equals the FRR. This threshold is defined as the Equal Error Threshold(EET) and the probability of error this threshold is called the Equal Error Rate or EER. The EET is then, the point of intersection of the Cumulative Distribution Functions of both classes as illustrated by the figure on the right hand side of Figure 5.2. The EER is used as a performance metric, in order to evaluate the effectiveness of the Speaker Verification system, in this thesis.

5.7 Model Training and Bayesian Adaptation

The standard approach for creating speaker models, both individual target speaker models and composite speaker background models, makes use of the expectation-maximization (EM) algorithm in the Maximum Likelihood framework applied to the training data. We refer to such individually trained speaker models as "scratch" models. A second approach for creating individual target speaker models is to adapt them from a so-called universal background model (UBM) by means of a form of Bayesian adaptation [4, 32]. This method of creating a target speaker models the falls under the MAP estimation framework using the EM.

5.7.1 The Auxiliary Function and Bayesian Adaptation

The auxiliary function for MAP estimation can be stated as [32, 34]:

$$\mathbf{R}(\mathbf{\Lambda}, \mathbf{\Lambda}^{'}) = \mathbf{Q}(\mathbf{\Lambda}, \mathbf{\Lambda}^{'}) + \log p(\mathbf{\Lambda}) \tag{5.26}$$

where $\mathbf{Q}(\mathbf{\Lambda}, \mathbf{\Lambda}^{'})$ is the auxiliary function stated in Section 5.5.1 for the EM Algorithm in an ML framework and $p(\mathbf{\Lambda})$ is the prior distribution over the respective parameters of Gaussian Mixture density, and is equivalent to the prior distribution $g(\theta)$ stated in Section 5.4. Although a discussion about the nature of these prior distributions over each parameter is beyond the scope this thesis, Gauvain et al. discuss the topic of suitable prior distributions in [32] . Further, with certain constraints imposed on the distributions, a form of Bayesian Adaptation can be employed for adapting the parameters of a speaker model from a UBM [4, 32].

Following Reynolds in [4], given a UBM with parameters $\mathbf{\Lambda} \equiv [\mu_k, \mathbf{\Sigma}_k, \pi_k]\ k = 1, \ldots, K$ the mean parameter μ_k can be adapted to an individual's training data using the formulation,

$$\mu_k^{'} = \alpha_k E_k(\mathbf{X}) + (1 - \alpha_k)\mu_k \tag{5.27}$$

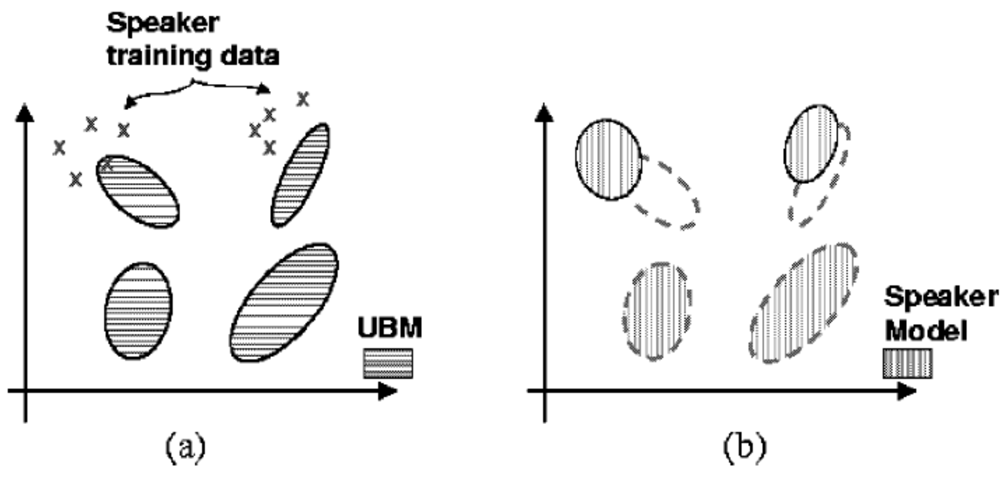


Figure 5.3: Pictorial example of two steps in adapting a hypothesized speaker model. (a)The training vectors ($\mathbf{x}$'s) are probabilistically mapped into the UBM mixtures (b) The adapted mixture parameters are derived using the statistics of the new data and UBM mixture parameters. The adaptation is data dependent, so UBM mixture parameters are adapted by different amounts. Taken from [4]

where,

$$E_k(\mathbf{X}) = \frac{1}{n_k}\sum_{t=1}^{T} Pr(k|\mathbf{x}_t)\mathbf{x}_t \tag{5.28}$$

$$Pr(k|\mathbf{x}_t) = \frac{\pi_k \mathcal{N}(\mathbf{x}_t|\mu_k, \mathbf{\Sigma}_k)}{\sum_{l=1}^{K}\pi_l \mathcal{N}(\mathbf{x}_t|\mu_l, \mathbf{\Sigma}_l)} \quad t = 1, 2, \ldots, T \tag{5.29}$$

and

$$n_k = \sum_{t=1}^{T} Pr(k|\mathbf{x}_t) \tag{5.30}$$

The adaptation coefficient, α_k, controls the relative weight of the training data for updating the model parameter, μ_k. A typical value for α_k is given by,

$$\alpha_k = \frac{n_k}{n_k + r} \tag{5.31}$$

where r is a fixed, so-called, relevance factor. As stated in [4], the relevance factor is generally set to a value of 16.

The first step in the Bayesian Adaptation process is similar to the E-step of the EM Algorithm for estimating the parameters of the GMM in a ML framework. This involves calculating estimates of the sufficient statistics of the speaker training data computed for each mixture of the UBM. These sufficient statistics are calculated as in

Eqs. (5.28), (5.29), and (5.30). The new sufficient statistics thus calculated are used to adapt parameters of the mixtures of the UBM as is done in Eq. (5.27) for the mean parameter. Although other parameters, π_k and $\mathbf{\Sigma}_k$ can be similarly adapted, in the experiments carried out as a part of this thesis, only the set of mean vectors, μ_k, were adapted. Fig. 5.3, illustrates the two steps of the process of Bayesian Adaptation. After a single Bayesian Adaptation step of creating a target speaker model by adapting from the UBM, the process of adaptation can continued to be performed iteratively, by substituting $\mu_k^{'}$ for μ_k in each iteration. In the experiments performed here, the performance of target speaker models obtained by, both, a single Bayesian Adaptation step and a single Bayesian Adaptation step followed by four iterations of the same, were carried out and compared. It is also useful to mention here that to gauge how "far" the adapted target speaker model has moved from the UBM, the Bhattacharya distance [37], between the component Gaussians of the UBM and the adapted speaker model can be calculated.

Chapter 6

Experimental Setup

6.1 Experimental Setup for Speech Verification

6.1.1 An Overview of the Software Utilities

Version 3.7 of the Sphinx Speech Recognition System [38] from Carnegie Mellon University(CMU)(henceforth referred to simply as Sphinx) was the Automatic Speech Recognition system that was used to develop the Speech Verification system. The Sphinx package consisted of a set of programs which could be used to:

- train speaker independent acoustic models using continuous mixture density Hidden Markov Models (HMMs)
- force-align transcriptions to a sequence of acoustic models representative of the linguistic sounds within the transcription
- decode and recognize unknown speech utterances using a given (trained) set of acoustic and language models.

All of these programs were used during the training and the task evaluation stages. Sphinx did not come with its own Language Model(LM) training utility, and therefore an external LM training toolkit (called the CMU-Cambridge Statistical Language Modeling (SLM) Toolkit [39]) was used.

Training data from the CMU Communicator Project Database was used for training the acoustic and language models. Speech utterances and their transcriptions from the CMU Communicator Project Database were also used in the speech verification experiments.

```
1.I'D LIKE TO(2) GO TO(3) WASHINGTON  IN(2) THE(3) EVENING
2.TOMORROW TOMORROW IN(2) THE(3) EVENING
3.YEAH THAT ONE'S GREAT
4.NO
5.I'D LIKE TO(2) GO TO PENSACOLA_FLORIDA
6.I'D LIKE TO(3) LEAVE TOMORROW IN THE(3) EVENING
7.I'D LIKE TO(3) LEAVE ON AUGUST EIGHTEENTH
8.IN(2) THE(3) AFTERNOON YEAH THAT ONE'S GREAT
9.THIS IS NOT OKAY NO
10.YEAH THAT ONE'S GREAT
```

Figure 6.1: Sample Transcriptions of Speech utterances from the Communicator Corpus

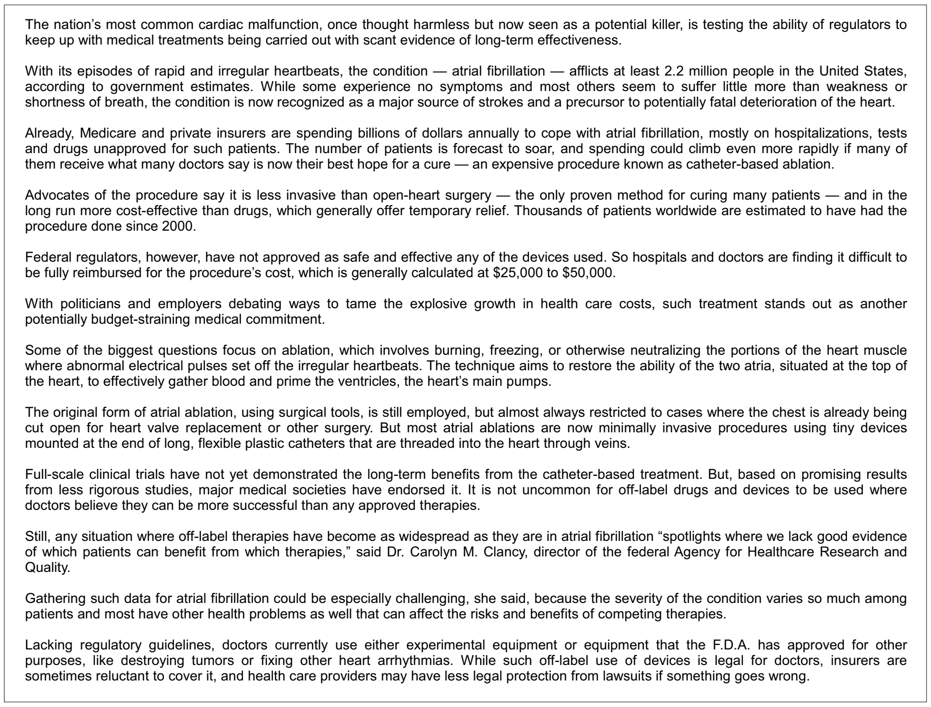
6.1.2 The CMU Communicator Corpus

The CMU Communicator Corpus was created as a part of the data collection phase of the ATIS (Air Travel Information System) project which was funded by the DARPA Communicator program [40]. The resulting database consists of approximately 90 hours of transcribed speech data, containing 180,605 speech utterances. The data was collected from a large population of callers (approx. 2000) who called into the CMU Travel Planning System. The speech data was sampled at 8 Khz and quantized using 16 bit samples. The entire data-set includes 10,417 distinct words. Apart from accurate word level transcriptions, the transcriptions include labels for acoustic events of importance including environmental noise, human-generated noise, system intrusions, and meta-tags which flag asides and comments. A set of sample transcriptions of 10 speech utterances from the Communicator Corpus is shown in Fig. 6.1.

6.2 Experimental Setup for the Speaker Verification system

6.2.1 Data Collection for the Speaker Verification system evaluation

The speech samples used to evaluate the speaker verification system were recorded over dialed-up telephone lines using a THAT-1 Telephone Handset Audio Tap available from BK Audio Products. Recordings were made using an ordinary telephone handset in a quiet office environment. 42 male, native-born, speakers of English were recruited to participate in the recordings. Speakers provided either one or two recording sessions for a total of 59 sessions. When 2 sessions were provided, they were made on different days.



The nation’s most common cardiac malfunction, once thought harmless but now seen as a potential killer, is testing the ability of regulators to keep up with medical treatments being carried out with scant evidence of long-term effectiveness.

With its episodes of rapid and irregular heartbeats, the condition — atrial fibrillation — afflicts at least 2.2 million people in the United States, according to government estimates. While some experience no symptoms and most others seem to suffer little more than weakness or shortness of breath, the condition is now recognized as a major source of strokes and a precursor to potentially fatal deterioration of the heart.

Already, Medicare and private insurers are spending billions of dollars annually to cope with atrial fibrillation, mostly on hospitalizations, tests and drugs unapproved for such patients. The number of patients is forecast to soar, and spending could climb even more rapidly if many of them receive what many doctors say is now their best hope for a cure — an expensive procedure known as catheter-based ablation.

Advocates of the procedure say it is less invasive than open-heart surgery — the only proven method for curing many patients — and in the long run more cost-effective than drugs, which generally offer temporary relief. Thousands of patients worldwide are estimated to have had the procedure done since 2000.

Federal regulators, however, have not approved as safe and effective any of the devices used. So hospitals and doctors are finding it difficult to be fully reimbursed for the procedure’s cost, which is generally calculated at $25,000 to $50,000.

With politicians and employers debating ways to tame the explosive growth in health care costs, such treatment stands out as another potentially budget-straining medical commitment.

Some of the biggest questions focus on ablation, which involves burning, freezing, or otherwise neutralizing the portions of the heart muscle where abnormal electrical pulses set off the irregular heartbeats. The technique aims to restore the ability of the two atria, situated at the top of the heart, to effectively gather blood and prime the ventricles, the heart’s main pumps.

The original form of atrial ablation, using surgical tools, is still employed, but almost always restricted to cases where the chest is already being cut open for heart valve replacement or other surgery. But most atrial ablations are now minimally invasive procedures using tiny devices mounted at the end of long, flexible plastic catheters that are threaded into the heart through veins.

Full-scale clinical trials have not yet demonstrated the long-term benefits from the catheter-based treatment. But, based on promising results from less rigorous studies, major medical societies have endorsed it. It is not uncommon for off-label drugs and devices to be used where doctors believe they can be more successful than any approved therapies.

Still, any situation where off-label therapies have become as widespread as they are in atrial fibrillation “spotlights where we lack good evidence of which patients can benefit from which therapies,” said Dr. Carolyn M. Clancy, director of the federal Agency for Healthcare Research and Quality.

Gathering such data for atrial fibrillation could be especially challenging, she said, because the severity of the condition varies so much among patients and most have other health problems as well that can affect the risks and benefits of competing therapies.

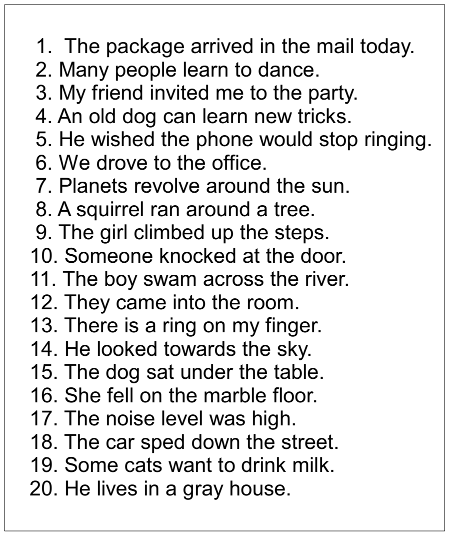
Lacking regulatory guidelines, doctors currently use either experimental equipment or equipment that the F.D.A. has approved for other purposes, like destroying tumors or fixing other heart arrhythmias. While such off-label use of devices is legal for doctors, insurers are sometimes reluctant to cover it, and health care providers may have less legal protection from lawsuits if something goes wrong.

Figure 6.2: Training Text

Each recording session was carried out using two types of read text. The first type was a newspaper story of about 4 minutes in duration, as shown in Fig. 6.2. The second type consisted of a list of 20 different 5- to 7-word long sentences that were automatically generated according to a supplied word grammar and were both grammatically and syntactically correct. An example of one such set of 20 different utterances is shown in Fig. 6.3. The 4-minute newspaper story was the same for all recording sessions. A different 20-sentence list was used for each speaker and recording session.

6.2.2 Speaker Verification Evaluation

The experimental speaker set was divided into two sets. There were 17 speakers who completed two recording sessions. These speakers were designated “customers” or target speakers. The remaining 25 speakers, who each completed just one recording session were designated “imposters” or false speakers. Target speaker models were constructed using the 4-minute news story recorded in the speaker’s first recording session. Target speaker test data was obtained from the 20-sentence list recorded in the speaker’s second



1. The package arrived in the mail today.
2. Many people learn to dance.
3. My friend invited me to the party.
4. An old dog can learn new tricks.
5. He wished the phone would stop ringing.
6. We drove to the office.
7. Planets revolve around the sun.
8. A squirrel ran around a tree.
9. The girl climbed up the steps.
10. Someone knocked at the door.
11. The boy swam across the river.
12. They came into the room.
13. There is a ring on my finger.
14. He looked towards the sky.
15. The dog sat under the table.
16. She fell on the marble floor.
17. The noise level was high.
18. The car sped down the street.
19. Some cats want to drink milk.
20. He lives in a gray house.

Figure 6.3: Example of a set of 20 testing phrases used in Speaker Verification Evaluation

recording session. Imposter test data consisted of all the 20-sentence lists recorded by the 25 1-session speakers.

In speaker verification, test utterances are compared both with the target model and a (jack-knifed) speaker background model. Speaker background models are used to normalize and stabilize test scores and improve performance. Speaker background models are ideally constructed from training data obtained from a large population of speakers with a range of texts and under conditions similar to those used for training target speakers. In this data collection there was an insufficient amount of data and number of speakers to provide an independent universal background model for all target speakers. Instead a different background model was constructed for each target speaker consisting of the pooled training data from the remaining 16 target speakers.

Verification performance was measured in the following way. For each target speaker, the 20 test sentences were compared with the target speaker and background speaker models to provide a set of “true” speaker scores. In addition, the 20 test sentences from each of the 25 imposters, for a total of 500 sentences, were also compared with the target speaker and background speaker models to provide a set of “false” speaker scores.

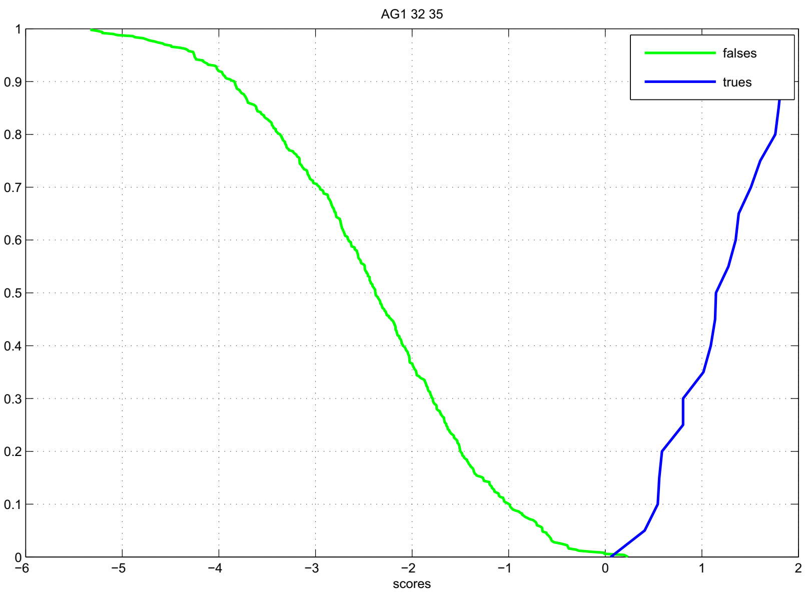


Figure 6.4: Empirical true and false cumulative probability distributions for an individual target speaker showing the equal-error threshold and rate

The true speaker and false speaker scores were sorted to provide empirical cumulative true and false probability distributions of the scores, an example of which is shown in Fig. 6.4. The value of the threshold at the intersection of the true and false score plots is the so-called equal-error threshold. At this threshold, the estimated probability of rejecting a true speaker test utterance is equal to the estimated probability of accepting a false speaker test utterance. The estimated equal-error probability was the figure of merit used for verification performance in this evaluation.

6.2.3 MATLAB Toolbox for Speaker Verification

Speaker Verification evaluations were initially carried out using a set of programs written in C and C++ from AT&T Labs. To further development and research of the Dynamic Pass-phrase Voice Security System, there was a requirement that a Speaker Verification system be developed at Rutgers. Therefore, as a part of this work, a new software implementation was written in MATLAB for the speaker verification evaluation task that was described in Section 6.2.2. This implementation was base-lined on

Spkr	EER(%)
WS1	3.6
CM1	0.6
JW1	0.0
JM1	20.0
JS1	2.0
IS1	7.2
JW2	1.2
KM1	0.0
AM2	10.0
AG1	0.6
MA1	5.0
NR1	5.0
LR1	0.0
AR1	0.0
OH1	18.6
JD1	19.4
DG1	40.00
avg/mdn	7.6/3.6

Table 6.1: Equal Error Rates in % for the 17 speakers modeled by 32 mixture GMMs, with feature vectors clipped at -35dB

results obtained from the AT&T Labs Speaker Verification system as shown in Fig. 6.5. Table 6.1 shows experimental values of the equal-error rates for the 17 customers along with the average(avg.) and the median(mdn.) error rates(The details of this run i.e., the number of mixtures, clipping level etc. are discussed in a later chapter.). To meet the requirement of training speaker models with reduced amounts of training data, additional routines in MATLAB were written to adapt target speaker models from a speaker background model.

The layout of the MATLAB system is shown in Fig. 6.6. A main control script, within which a number of system options can be set, controls the flow of execution. The control script chooses the front end implementations(AT&T Labs or the MATLAB Auditory Toolbox), chooses a target speaker(from the 17 customers) and a list of background speakers, and enables one of the following:

- training "scratch" speaker models
- training of speaker background models

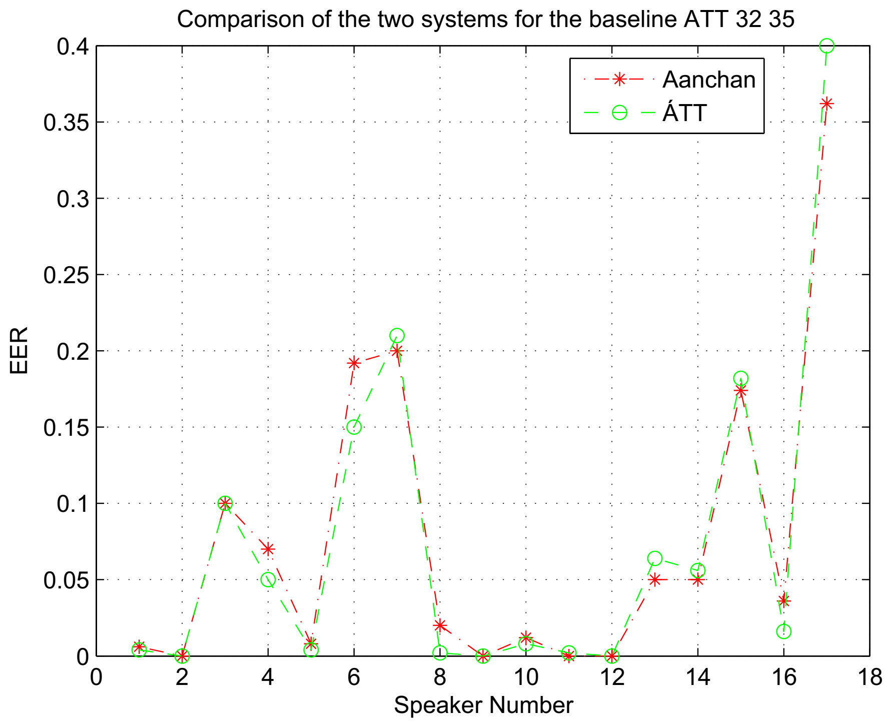


Figure 6.5: Comparison of the AT&T and the MATLAB systems based on the Equal Error Rates for each individual speaker(The EERs are normalized to 1, and are not presented in % as mentioned in Table 6.1.). The speech vectors for training are clipped at an energy threshold of -35dB, with 32-mixture GMMs used for target and background models.

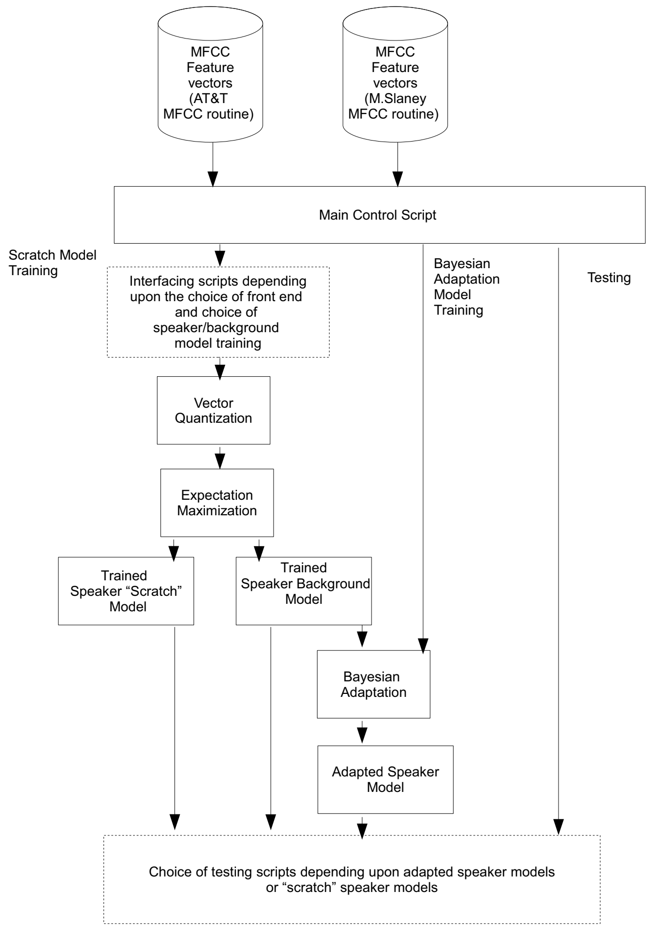

MFCC Feature vectors (AT&T MFCC routine)
MFCC Feature vectors (M.Slaney MFCC routine)
Main Control Script
Scratch Model Training
Bayesian Adaptation Model Training
Testing
Interfacing scripts depending upon the choice of front end and choice of speaker/background model training
Vector Quantization
Expectation Maximization
Trained Speaker "Scratch" Model
Trained Speaker Background Model
Bayesian Adaptation
Adapted Speaker Model
Choice of testing scripts depending upon adapted speaker models or "scratch" speaker models

Figure 6.6: MATLAB Speaker Verification System Layout

- training of adapted speaker models
- testing and evaluating performance for a given set of verification system parameters

by calling on various scripts depending on the options that are chosen by the user.

Chapter 7
Speech Verification Experiment

7.1 Training and Preparation

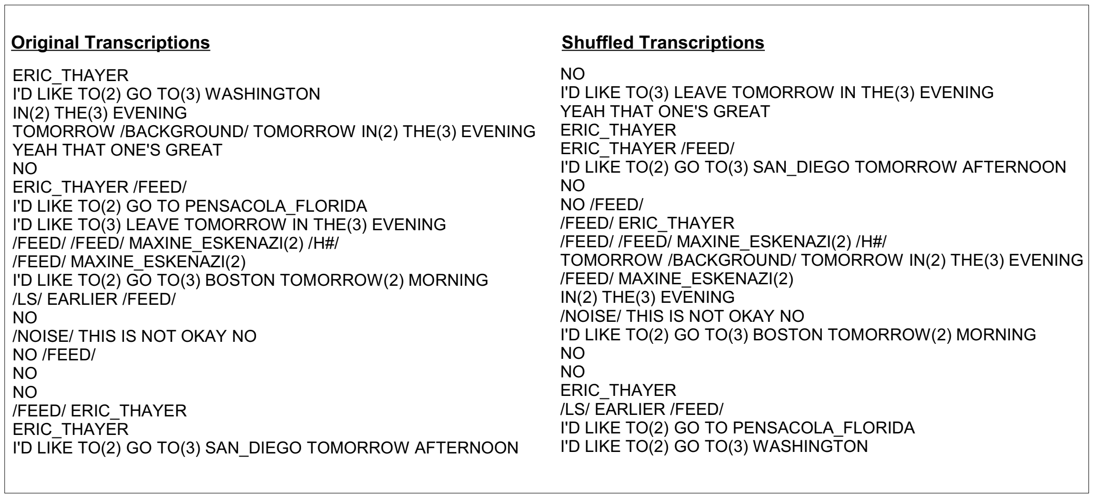
7.1.1 Training the Acoustic and Language Models

The acoustic models provided by CMU were trained on a subset of the CMU Communicator Corpus, which consisted of 124,760 utterances with a training dictionary of 9757 words and 55 phones. The feature vector for each frame of speech consisted of a set of 13 mel-frequency cepstral, 13 delta mel-frequency cepstral and 13 delta-delta mel-frequency cepstral coefficients, for a total of 39 features per frame. The acoustic models were trained using three state HMMs with 32, 64 or 128 Gaussian mixtures per state, and 2000 tied states. For the speech verification experiments discussed in this thesis, acoustic models trained with 32 Gaussian mixtures per state were used.

The language model was trained using the Statistical Language Modeling(SLM) Toolkit. The procedure for training a language model is provided in [40]. A traditional tri-gram language model (using a backoff strategy for low count tri-grams) was created using the given transcriptions and the dictionary provided with the CMU Communicator Corpus. Care was taken to remove filler words and words that were used to describe events in the transcription like /CLICK/ or /FEED/. Once the transcriptions were stripped of these words, there remained 117,429 utterances in the reduced training set.

7.1.2 Data preparation for the Speech Verification Experiments

A subset of 3000 utterances out of the 117,429 speech utterances, whose transcriptions were used to train the language model, were used in the speech verification experiments.



Original Transcriptions	Shuffled Transcriptions
ERIC_THAYER	NO
I'D LIKE TO(2) GO TO(3) WASHINGTON	I'D LIKE TO(3) LEAVE TOMORROW IN THE(3) EVENING
IN(2) THE(3) EVENING	YEAH THAT ONE'S GREAT
TOMORROW /BACKGROUND/ TOMORROW IN(2) THE(3) EVENING	ERIC_THAYER
YEAH THAT ONE'S GREAT	ERIC_THAYER /FEED/
NO	I'D LIKE TO(2) GO TO(3) SAN_DIEGO TOMORROW AFTERNOON
ERIC_THAYER /FEED/	NO
I'D LIKE TO(2) GO TO PENSACOLA_FLORIDA	NO /FEED/
I'D LIKE TO(3) LEAVE TOMORROW IN THE(3) EVENING	/FEED/ ERIC_THAYER
/FEED/ /FEED/ MAXINE_ESKENAZI(2) /H#/	/FEED/ /FEED/ MAXINE_ESKENAZI(2) /H#/
/FEED/ MAXINE_ESKENAZI(2)	TOMORROW /BACKGROUND/ TOMORROW IN(2) THE(3) EVENING
I'D LIKE TO(2) GO TO(3) BOSTON TOMORROW(2) MORNING	/FEED/ MAXINE_ESKENAZI(2)
/LS/ EARLIER /FEED/	IN(2) THE(3) EVENING
NO	/NOISE/ THIS IS NOT OKAY NO
/NOISE/ THIS IS NOT OKAY NO	I'D LIKE TO(2) GO TO(3) BOSTON TOMORROW(2) MORNING
NO /FEED/	NO
NO	NO
NO	ERIC_THAYER
/FEED/ ERIC_THAYER	/LS/ EARLIER /FEED/
ERIC_THAYER	I'D LIKE TO(2) GO TO PENSACOLA_FLORIDA
I'D LIKE TO(2) GO TO(3) SAN_DIEGO TOMORROW AFTERNOON	I'D LIKE TO(2) GO TO(3) WASHINGTON

Figure 7.1: Original Transcriptions and Shuffled Transcriptions

The Sphinx setup uses a text file, called a "transcription" file to store transcriptions of all the speech utterances. A "control" file contains the names of the files of the speech utterances to which each transcription in the "transcription" file corresponds, thereby providing a one-to-one mapping between a sentence level transcription and its corresponding speech file. For the experiments performed in this research two subsets of 3000 utterances were created, one in which the correspondence between sentence level transcriptions and their speech files were maintained intact, and a second set of transcriptions in which there was no correspondence between a sentence-level transcription and its speech utterance. The first set of transcriptions were called "original" transcriptions, and the second set was called the set of "shuffled" transcriptions.

The "shuffled" transcriptions were obtained by shuffling the set of "original" transcriptions by using the Fisher-Yates Shuffle Algorithm [41] implemented in PERL, which was taken from [42]. Fig. 7.1 shows examples of "original" and "shuffled" transcriptions of several utterances.

7.2 The Speech Verification Experiment

The set of transcriptions that was used for training the language model was also used for a speech verification experiment that is described in this section.

For the experiment two sets of transcriptions were created. The first set, which we call "original transcriptions", were transcriptions that were correctly matched to their corresponding speech utterances. Next, the "original transcriptions" were randomly shuffled at the sentence level to create a second set of transcriptions, which essentially removed the one-to-one correspondence between each speech utterance and its corresponding word-level transcription. We call this set the "shuffled transcriptions" set.

The first step in the speech verification experiment was to use the trained acoustic and language models to recognize a subset of 3000 out of the 117,429 speech utterances, in order to provide a set of recognition hypotheses, along with acoustic and language model likelihood scores for each word in each utterance. Only the acoustic model scores were used; the language model scores were discarded. Next, the "original transcriptions" were forced aligned with their corresponding speech utterances, to give a set of what we call "original forced-aligned acoustic likelihood scores". In the same manner the "shuffled transcriptions" were forced aligned against the same set of utterances to give "shuffled forced-aligned acoustic likelihood scores".

Intuitively if an incorrect transcription is forced aligned against an utterance, the acoustic likelihood score for that transcription should differ greatly from the score for the correct word transcription. The best acoustic word score that can be achieved occurs when the recognizer gets to consider all possible word sequences; however during forced alignment all that the recognizer had to work with was the transcription that was provided. There was no formal language model used during forced-alignment. Therefore following this rationale, we argue that the difference between the acoustic recognition score for an unconstrained utterance and for forced alignment of the same utterance against its correct transcription should essentially be zero. Similarly the difference between the unconstrained recognition score and the forced alignment score to an improper word transcription should be much larger and should vary widely depending on the degree of word mismatch in the shuffled transcriptions. Therefore if we create a plot of the distributions of these two difference scores, for the first case we would expect a distribution that was essentially an impulse at zero, and for the second case we would expect a spread out distribution with a long tail.

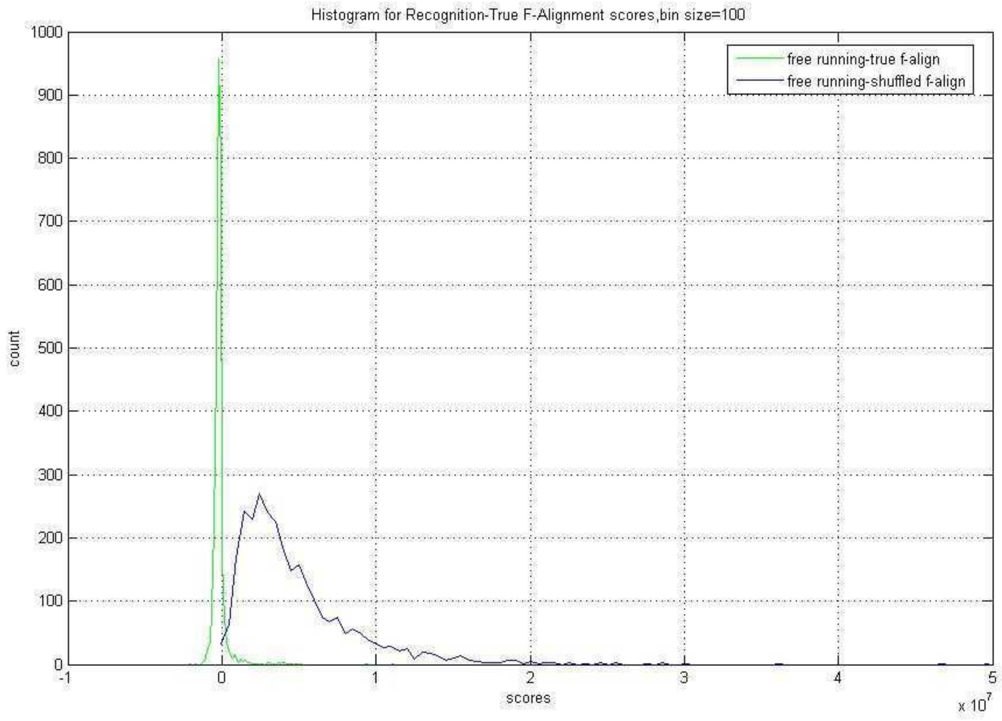


Figure 7.2: Plot of the Difference Scores

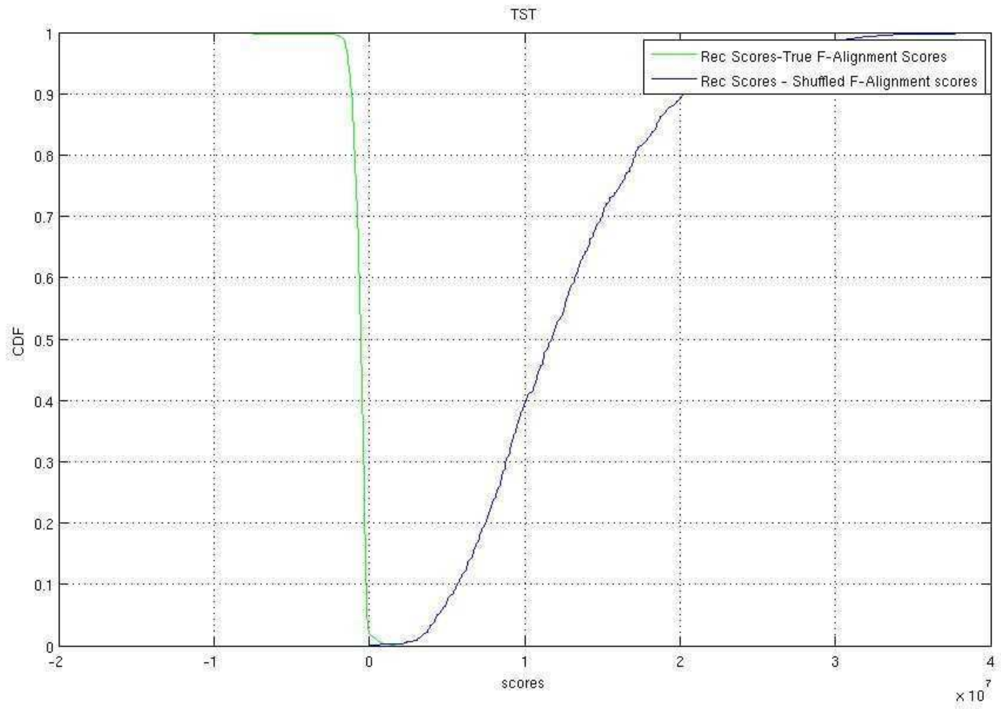


Figure 7.3: Plot of the CDF of the Difference Scores

Fig. 7.2 shows histogram plots of the measured difference scores for both sets of data. (These histogram plots essentially are estimates of the probability density functions of the difference scores for each of the two sets of data.) The distribution of the unconstrained recognition scores minus the constrained recognition score for the correctly transcribed utterance is shown in Fig. 7.2 and is seen to essentially be an impulse around zero value. The distribution of the unconstrained recognition score minus the constrained recognition score for an incorrectly transcribed utterance is the second histogram in Fig. 7.2 and is seen to have a mean that is significantly above zero and a large variance (as compared to the variance of the first distribution).

The point at which the two distributions of Fig. 7.2 intersect is called the Equal Error Threshold (EET) and the integrated value of the area under the curve from the Equal Error threshold to infinity (for the impulse-like distribution) is called the Equal Error Rate (EER). In order to estimate the EET, it is convenient to plot the cumulative distribution function for the two classes, as shown in Figure 7.3. It can be seen in this figure that the EET is slightly above zero, and the EER is a small fraction of 1%; i.e., there is virtually no overlap between the two distributions.

7.3 Conclusion

Based on the results of Figs. 7.2 and 7.3 it can be seen that the Sphinx Speech Recognizer can reliably (and essentially with no error) distinguish spoken sentences that do not exactly match the provided transcription from those that do exactly match the provided transcription. Hence it can be concluded that the speech verification engine performed its desired task well, namely guaranteeing that the speaker spoke the requested sentence, and with negligibly small error rate.

Chapter 8

Speaker Verification Experiments

8.1 Preliminaries

Before discussing performance of the speaker verification system, we begin with a discussion of convergence issues of the training algorithms, namely the Vector Quantization and the Expectation Maximization algorithms that were discussed in Chapter 5. Further it is well known that training large speaker background models is a challenging task since the EM algorithm is known for its high space and time complexity. Therefore, we propose and show the properties of a simple decimation technique which significantly speeds up the training process, with essentially no loss in accuracy of the resulting speaker verification system.

8.1.1 Convergence of the VQ and EM Algorithms

Figs. 8.1-8.3 illustrate the convergence properties of the VQ and EM Algorithms that were implemented as a part of the MATLAB Toolbox that was discussed in Section 6.2.3. Fig. 8.1 shows how the global average distortion(of the VQ design method) is reduced as the code-book size increases. Fig. 8.2 shows the convergence of the VQ global average distortion to within a specified threshold when a 64-element VQ code-book is created from a 32-element VQ code-book after a binary split. Finally Fig. 8.3 shows the convergence of the value of the log-likelihood of the model being estimated(via the EM method) with increasing number of iterations. Though the figure does not explicitly show convergence of the log-likelihood within the threshold, such convergence is guaranteed by increasing the number of iterations beyond 5. The performance gained by running the EM method for more than 5 iterations is negligible, and thus all results presented here are based on 5 iterations of the EM method.

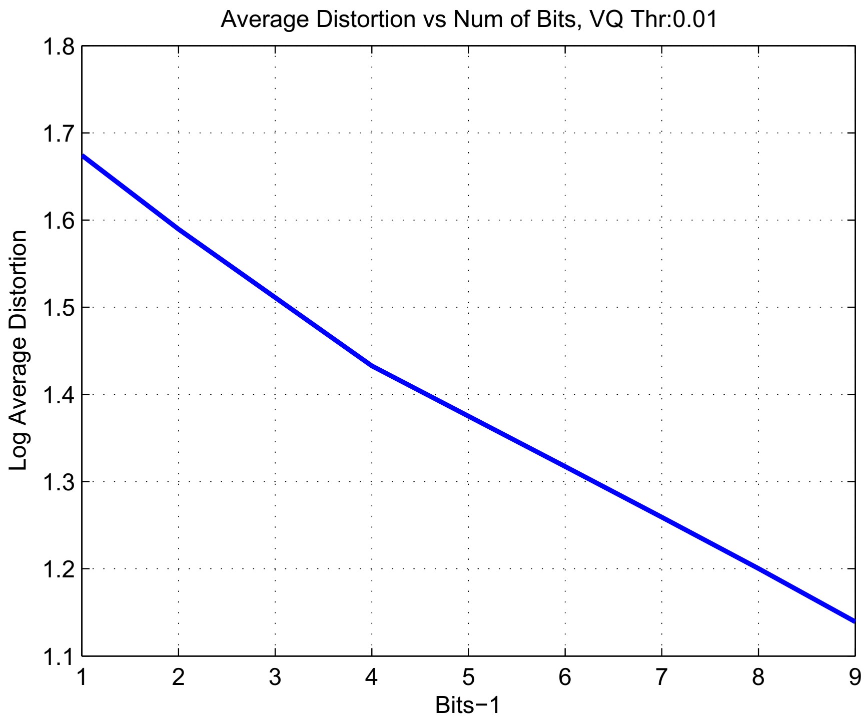


Figure 8.1: A plot showing the decreasing log-average distortion as a function of the number of bits needed to represent the number of elements in the code-book as a power of 2.

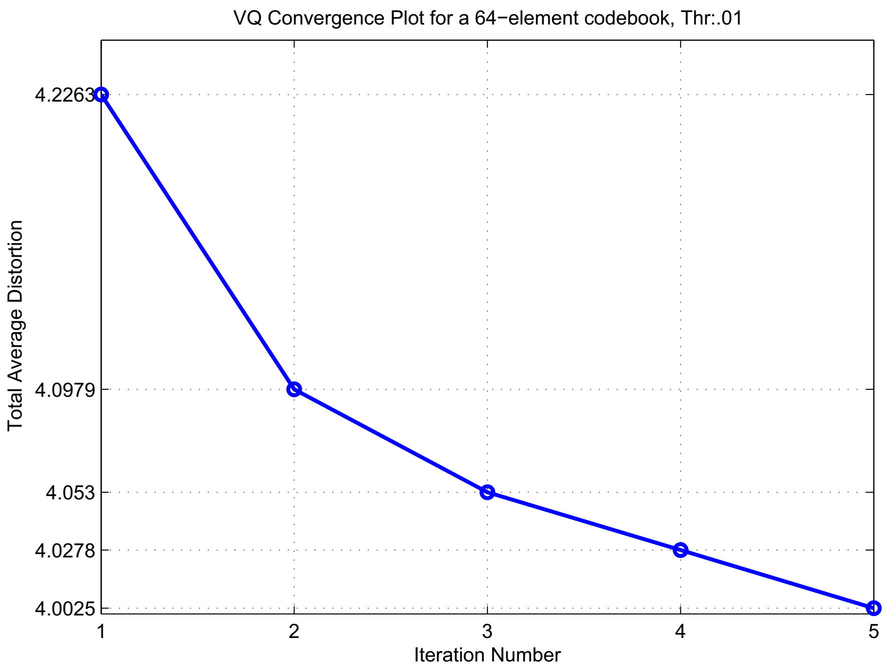


Figure 8.2: A plot of the average distortion showing the convergence of VQ Algorithm, while creating a 64-element code-book after a binary split on the corresponding 32-element code-book.

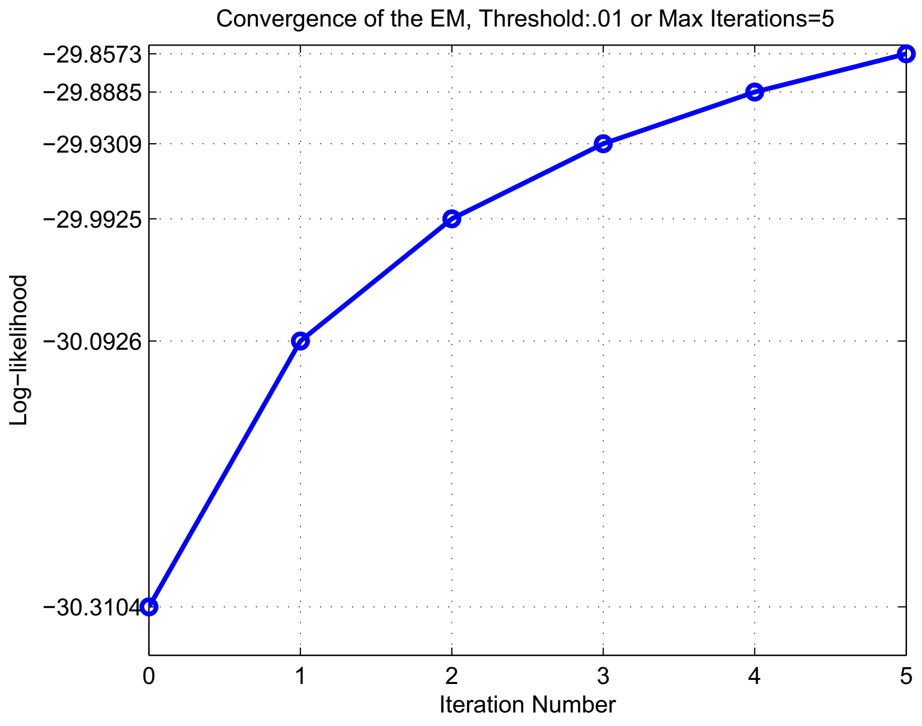


Figure 8.3: A plot of the log-likelihood showing the convergence of the EM Algorithm, for a 64-mixture GMM, where the starting point was the VQ code-book whose convergence plot is shown in Fig. 8.2.

8.1.2 A method to speed up background model training

Training speaker background models with large amounts of training data(order of 384,000 vectors) is quite a time consuming and memory intensive task. In fact, it is virtually impossible to run efficient vectorized implementations of training algorithms in MATLAB, while using large amounts of training data to train background models with the order of $K \geq 128$ mixtures, due to insufficient virtual memory.

In training speaker background models, the training data consists of speech pooled from a large number of speakers(16 in our experiments). In speaker verification tasks, such as the one described in Chapter 6, where the enrollment data consists of large amounts of read text(order of 4 minutes/talker), neighbouring speech frames from the training data of a single background speaker(or an impostor), tend to be highly correlated. Taking advantage of the large amount of data pooled in from various background speakers and the high degree of correlation among adjacent speech frames, the process of training background models can be significantly speeded up by down-sampling the speech frames used from each background speaker's training data. This process of down-sampling, which is called decimation, can be employed by selecting one frame out of every N frames or in other words by decimating the number of training frames from a background speaker's training data by a factor of $N : 1$. The decimation factor N is empirically determined for a given training set and depends on the number of mixtures being employed to train the background model. The performance loss is negligible, with a large reduction in the time required to train a background model. For the given speaker verification database, based on experimentation, the decimation factor N was chosen as given in Table 8.1. For training models with up to 128 mixture components, a decimation factor of 32 is appropriate reducing the training set size to about 12,000 vectors, or about 120 vectors per mixture on average. For training models with more than 128 mixture components, the decimation factor is reduced to keep about 120 vectors per mixture; hence we only decimate by a factor of 4 for 1024 mixture models.

To see the effect that decimation has on the VQ procedure, it is illustrative to look at plots of the variation of the global log-average distortion as the number of elements

Number of Mixtures	Decimation factor N
2-128	32
256	16
512	8
1024	4

Table 8.1: Decimation Factors that are chosen depending upon the number of mixtures in the background model

	64 mix.	128 mix.	256 mix.	512 mix.
Decimation	41.04 s	180.95 s	520.58s	1386.22 s
No Decimation	1628.34 s	6369.3 s	13503.5 s	$\geq$25000 s

Table 8.2: Comparison of the approximate training time in seconds for background models with increasing number of mixtures. It can clearly be seen that employing decimation greatly speeds up training with little or no loss in accuracy.

in the VQ code-books increase. Fig. 8.4 and Fig. 8.5 show that nothing is gained by retaining a larger number of highly correlated frames, and that the process of decimation has little or no effect on the final computed code-book vectors of the VQ-codebook.

Further to see the effect that the decimation process has on the EM procedure it is illustrative to look at the convergence of the log-likelihood value both with and without decimation being employed. Fig. 8.6 shows one such plot. The figure shows that though the two values of the likelihoods are numerically fairly close to one another, the value of the likelihood when decimation is used is slightly higher than when using the full training set(i.e., without decimation). This is true in almost all instances where decimation is employed.

Thus, the process of decimation yields a model that is very close to the actual one and yields performance that is very similar to a background model that is trained without decimation. The real advantage of decimation though can only be appreciated if one looks at the comparative values of the amount of time that is needed to train a background model both with and without decimation. Table 8.2 shows that the amount of time that is saved in training background models by employing this simple process of decimation is substantial.

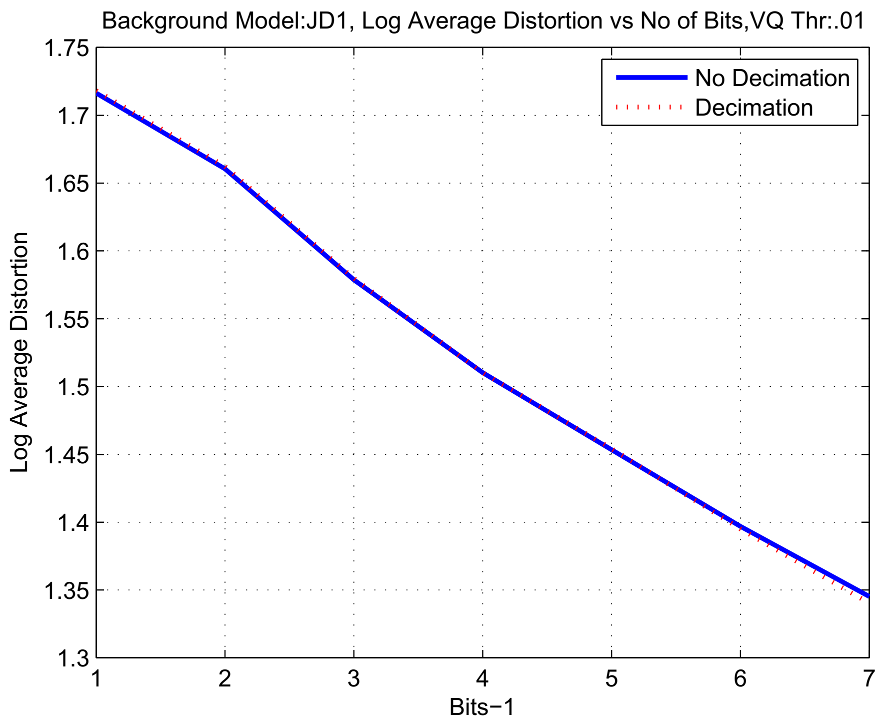


Figure 8.4: The variation of the global log-average distortion for a 128-element VQ-codebook, with and without decimation. The decimation factor N, employed here is 32. Hence only $\frac{1}{32}$ of the original number of frames is needed while training with decimation.

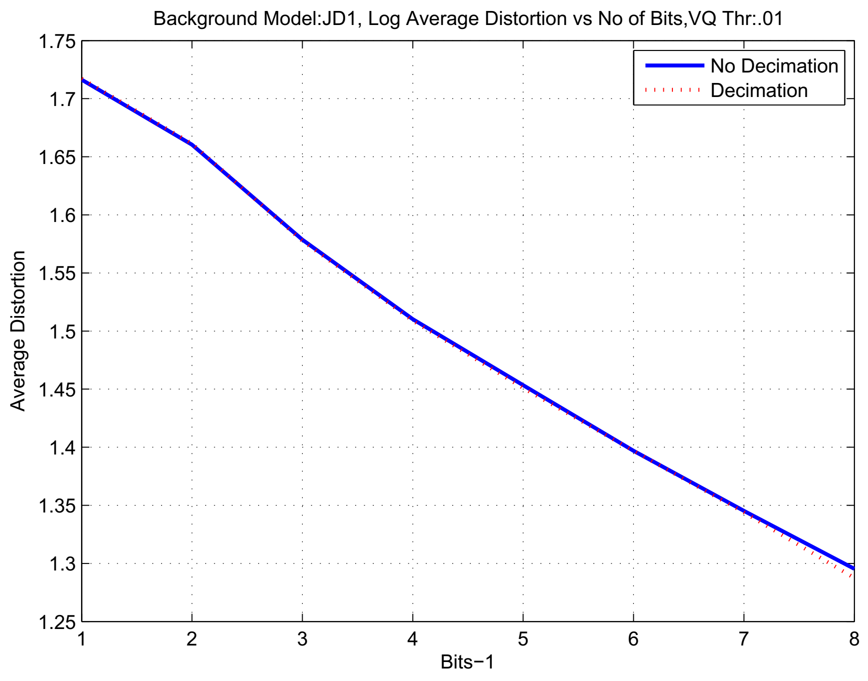


Figure 8.5: Similar to Fig. 8.4, the plot shows the variation of the global log-average distortion for a 256-element VQ-codebook.

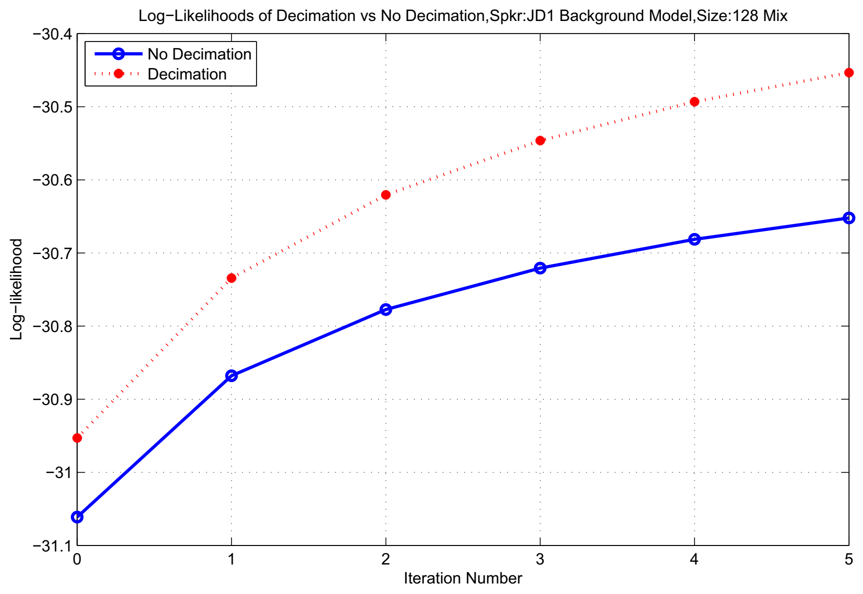


Figure 8.6: The EM procedure with and without decimation employed, for a GMM having 128 mixtures

8.2 Speaker Verification Experiments

A series of experiments was undertaken to evaluate the performance of the speaker verification system of Section 6.2.3. As described in Section 6.2.2 the figure of merit for verification performance is the equal-error rate(EER). The set of speaker verification experiments and their results are presented in three sets in this section.

First, the performances results are presented for both front-end implementations that were described in Chapter 4. The goal of this first step was to determine which of the two front-end implementations better suited the speaker verification task at hand. Next the impact of using different configurations of MFCC feature vectors on the speaker verification task was studied. Following this, the next set of experiments show the effect of reducing the amount of training data on the equal-error rate performance. It will also be shown that there is an advantage of using adapted speaker models over scratch models for limited amounts of training data. The results of all experiments are shown in the form of the variation of mean equal-error rate as the the number of mixture components in the GMMs vary from a small number of components to a large number of components. The last set of experiments investigate the performance impact of:

- using longer test utterances
- using two-sessions of training data for scratch models
- a technique called variance limiting on scratch models trained with limited amounts of training data

The baseline results were already presented in Table 6.1 of Section 6.2.3 in the context of comparing the MATLAB and the AT&T speaker verification systems. Table 6.1 shows individual equal error rate performance for models with 32-component GMMs. The features used were 12 MFCC coefficients appended with 12Δ MFCCs. An energy clipping level of -35dB was used, where speech frames whose relative energy fell below this threshold level were clipped or discarded. The mean and the median equal-error rates for the 17 customers are shown in the last row. It can be seen that there is a large

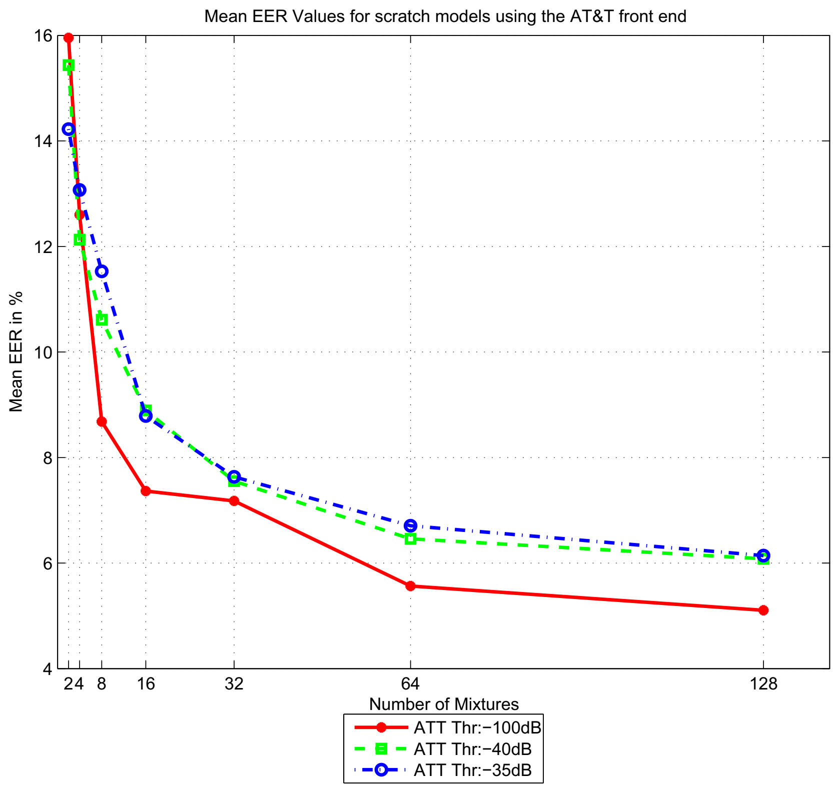


Figure 8.7: Mean EER values for models of size 2-128 mixtures, at clipping levels of -35dB,-40dB and -100dB, for the MFCC implementation from AT&T Bell Labs

variation in individual speaker performance. The mean equal-error rate was 7.6% while the median was 3.6%. For one speaker, DG1, the equal error rate was 40.0% which was unacceptably large. Another 4 speakers have equal-error rates between 10% and 20%, while 8 speakers had equal-error rates less than 2%.

8.2.1 Experimental evaluation of the AT&T and the MATLAB Auditory Toolbox front-end implementations

Two different MFCC feature extraction implementations were studied as a part of this thesis. The details of the two different implementations(AT&T and the MATLAB Auditory Toolbox) were discussed in Chapter 4. The goal of the first experiment was to determine which of the two MFCC feature extraction implementations provided the highest performance for the database of Chapter 6. The key variables of the system

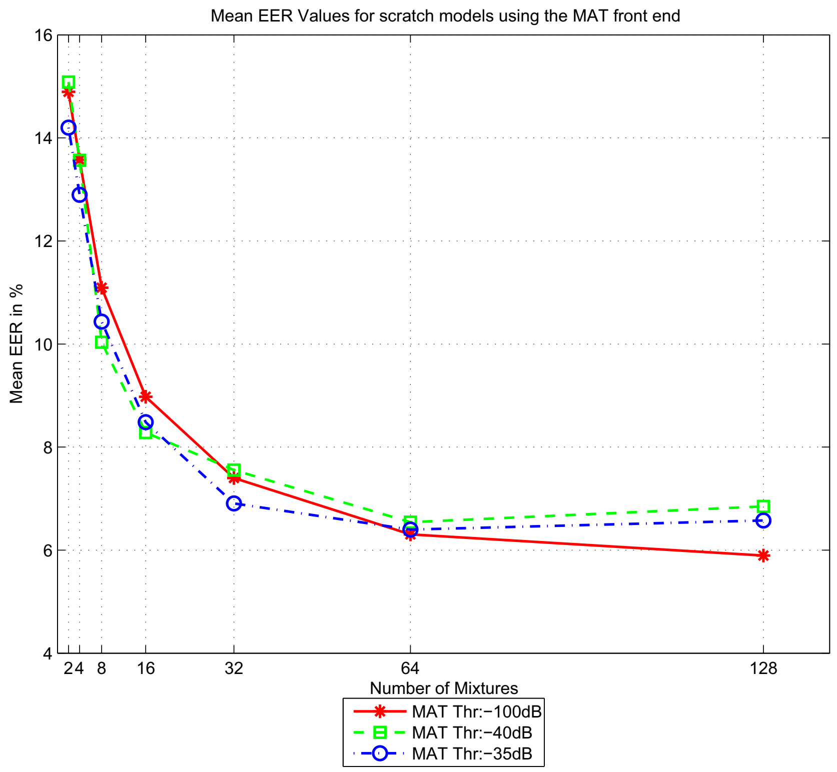


Figure 8.8: Mean EER values for models of size 2-128 mixtures, at clipping levels of -35dB,-40dB and -100dB, for the MFCC implementation from the MATLAB Auditory Toolbox

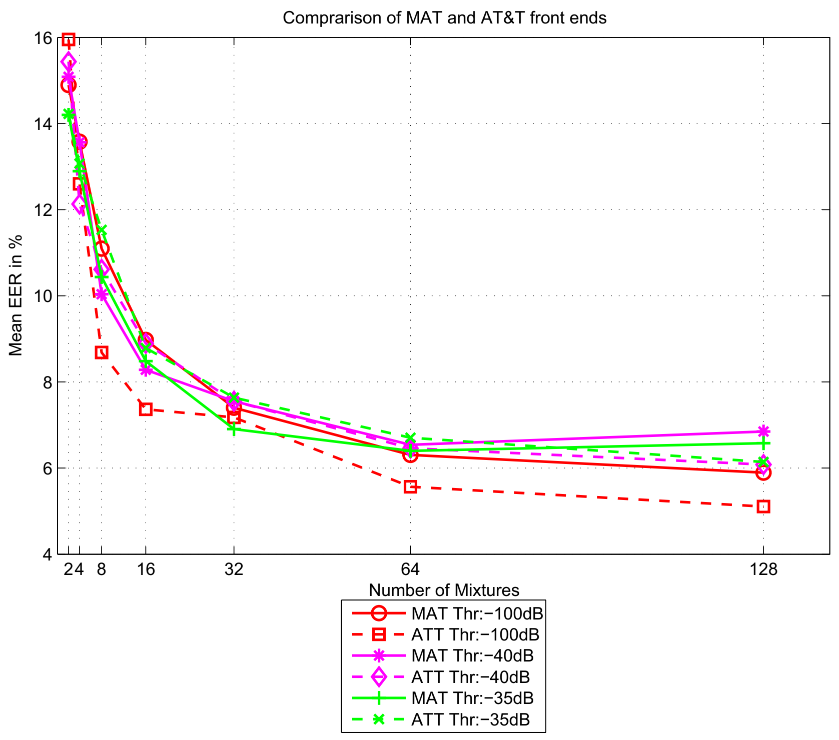


Figure 8.9: A combined plot of Figs. 8.7 and 8.8

Spkr	EER(%)
WS1	0.00
CM1	0.20
JW1	0.00
JM1	19.20
JS1	0.00
IS1	5.00
JW2	0.40
KM1	0.00
AM2	5.00
AG1	0.60
MA1	8.60
NR1	2.20
LR1	0.00
AR1	0.00
OH1	10.60
JD1	5.00
DG1	30.00
avg/mdn	5.11/0.6

Table 8.3: Equal-error rates in% for the individual speakers along with the average and the median equal-error rate in the last column. 12MFCC+12Δ MFCC features were extracted using the AT&T Front end implementation, with energy clipping maintained at a -100dB level. Each speaker is modeled by 128 mixture GMMs.

were the number of mixtures, K, in the GMM speaker and background models and the energy clipping level. The features used here again were 12 MFCCs appended with 12Δ MFCCs. For both front-end implementations, the variation of the mean equal-error rate is presented as the number of mixtures vary from 2-128. This experimental evaluation was repeated for three energy clipping levels namely -35dB, -40dB and -100dB. It must be noted that though the front-end implementations are different, the Gaussian Mixture Models trained as a result, used the same implementation (the MATLAB implementation discussed in 6.2.3 and 8.1.1) of the training algorithms (namely the VQ and the EM Algorithms). Fig. 8.7 displays these results for the AT&T Labs front-end implementation and Fig. 8.8 displays these results for the MATLAB Auditory Toolbox front-end implementation. Fig. 8.9 combines results from both these experiments.

Looking at both plots(Figs. 8.7 and 8.8) the first general trend one notices is that the performance steadily improves as the number of mixtures in the models increase. Usually choosing a model order(the number of mixtures K) is in general a non-trivial task and is heavily dependent on the nature and the amount of training data available. It is also dependent on the dimensionality and the nature of features used. It can, in general, be said that with sufficient training data, increasing the model order allows the model to capture a greater number of speech variations that occur in the training data. Increasing the model order beyond a point however tends to deteriorate performance, if there is insufficient training data to accurately fit the model. Higher model orders also cause a significant increase in the computational complexity for both training and testing.

Energy clipping is an important part of the the front-end processing (see Chapter 4). The source of low energy vectors is either background noise or low energy speech sounds. It is believed that eliminating these low-energy vectors from the training set of the speaker models would lead to performance improvements. However, a contrary point of view takes the position that including all data available in the recordings of speech samples leads to better models and performance. As stated in Chapter 4, low energy vectors are simply trimmed by omitting vectors whose log energy falls below a specified level relative to the peak log energy in sample utterances for both training and test utterances. As mentioned earlier, the effect of three clipping levels on the performance is investigated, namely -35dB, -40dB and -100dB. At a level of -35dB approximately 40% of the vectors are omitted, at -40dB approximately 30-35% of the vectors are omitted, while at -100dB no vectors are omitted.

From the performance plots for the AT&T front end implementation(Fig. 8.7) it can be seen that while the performance at higher trimming levels of -35dB and -40dB are comparable, the performance with no energy trimming(-100dB) is uniformly better. On the other hand from the performance plots for the MATLAB Auditory Toolbox front end implementation(Fig. 8.8) it can be seen for all energy trimming levels the performance is comparable for smaller size models($\leq$ 64 mixtures), while for larger size models($>$ 64 mixtures) the performance with no energy trimming(-100dB) is significantly better.

Overall, it can be argued that including all the training data for speaker modeling prove to be a definite advantage. It should be noted that both the training and testing data were of reasonably good quality and were hand endpointed. It is therefore highly likely that energy trimming might be more advantageous with lower quality data and less accurate endpointing.

To compare the performance between the two front-end implementations the performance plots for both front-ends are shown on a common plot in Fig. 8.9. Among all the different configurations of clipping levels and front-end implementations it can be seen that one configuration, namely the AT&T front-end implementation at a clipping energy level at -100dB emerges as the highest performance implementation. For this configuration the best performance was for models of size 128 mixtures. The individual speaker equal-error rates for this configuration are presented in Table 8.3. By examining the plots in Figs. 8.7-8.9 one can also draw the conclusion that speaker verification performance is, in fact, sensitive to the configuration of the filters in the Mel filter bank. The AT&T front end configuration is thus better tuned to extracting more perceptually important features that are vital to the speaker verification task.

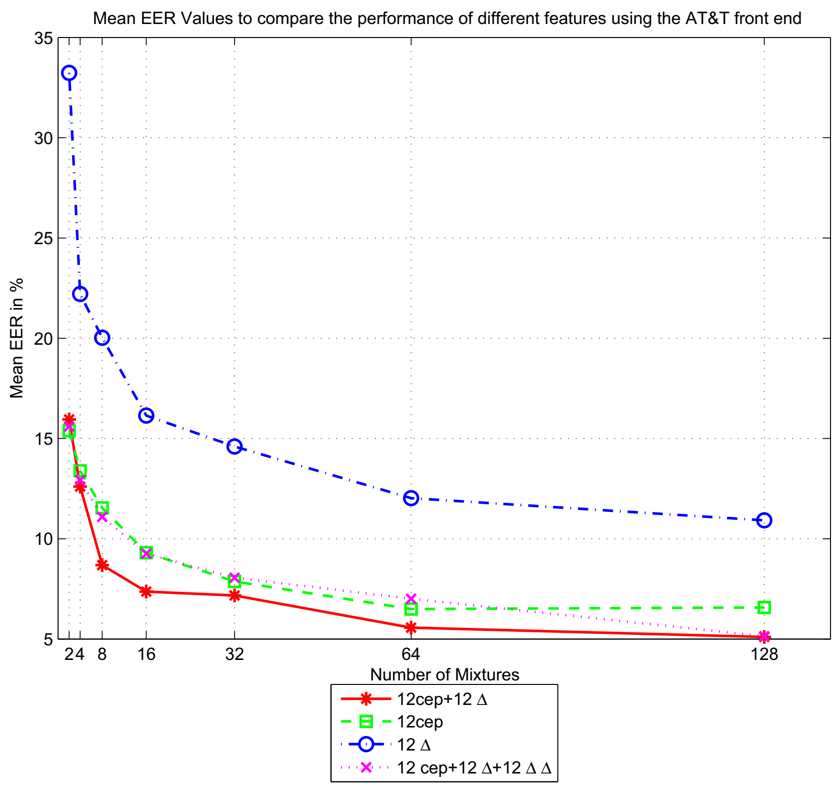


Figure 8.10: A comparison of the performance of different choices of features

8.2.2 Experiments to determine the best configuration of features

The next set of experiments deal with investigating the best configuration of MFCC features to use for the speaker verification task. The feature configurations explored are 12 MFCCs, 12MFCC+12Δ MFCC, 12MFCC+12Δ MFCC+12$\Delta\Delta$ MFCCs and 12Δ MFCCs. Following from the first set of experiments, an energy clipping level of -100dB was used, and the features were extracted using the AT&T front end implementation, for all the feature configurations explored. Also, the number of mixtures, K, is varied between 2-128 mixtures as was done for the first set of experiments, for all feature configurations.

The results of the experiments are shown in Fig. 8.10. The role and importance of using difference coefficients was discussed in Section 4.1. In the same section it was also mentioned that the difference coefficients alone do not perform as well as the static MFCC features. This can clearly be seen in the plot where the performance using only ΔMFCC vectors was significantly worse than any other feature vector choice. On the other hand the best overall performance is obtained by using 12 MFCC appended with the first difference coefficients. The performance of the two remaining feature sets(the MFCC alone and the MFCC+ΔMFCC+$\Delta\Delta$MFCC) was somewhat worse than the performance using MFCC+ΔMFCC over the range $2 \leq K \leq 64$ mixtures. For 128 mixtures however, the performance of the 12 MFCCs appended with the first and second differences seemed to match the performance of 12 MFCCs appended with the first difference coefficients.

By using features consisting of 12 MFCCs appended with first and second difference frames, we are essentially increasing the dimensionality to 36 features. Increasing the dimensionality is much of a debated issue in Pattern Recognition literature. While adding additional features may aid in incorporating extra information in order to boost performance, there are certain caveats. Higher dimensional features demand increased computational complexity. Further, higher dimensional features require the use of more complex models. It is clear from Fig. 8.10 that for the 36-dimensional features, the lower order and simpler models just cannot capture all the variations of the feature set in the higher dimensional feature space. As the number of mixtures increase, the GMMs start to better model the variations in the features in the higher dimensional space. It will be seen, in the subsequent experiments, that as the model size grows larger the performance using these higher dimensional features dramatically improves.

8.2.3 Experiments with reduced training and Bayesian Adaptation

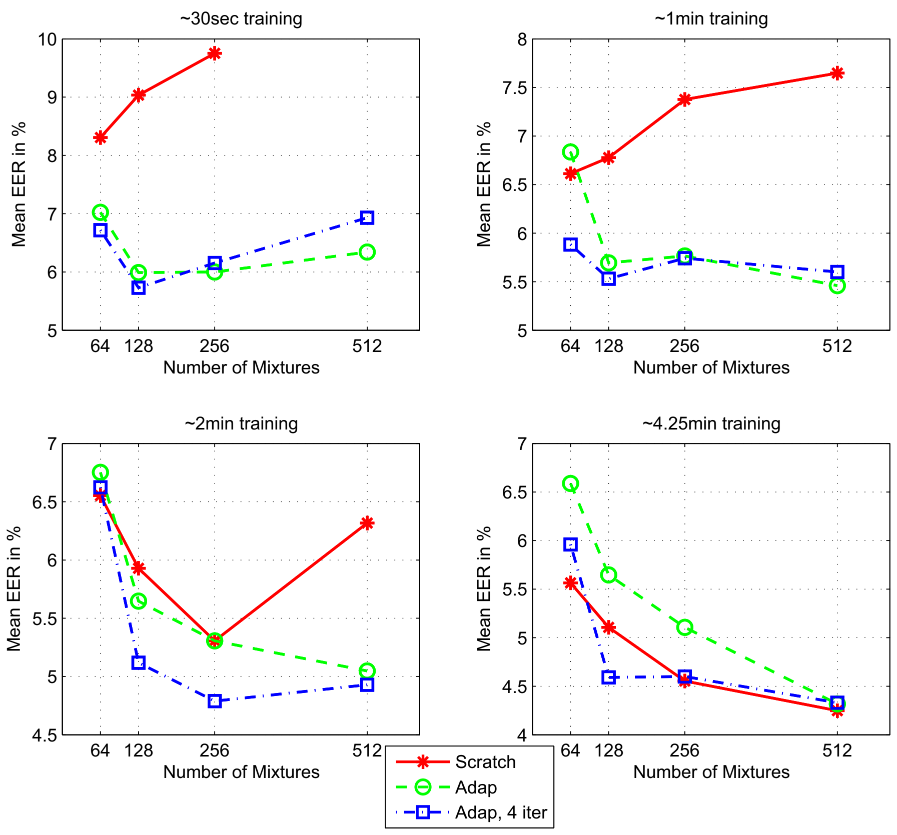


Figure 8.11: A comparison of the performance of Bayesian Adaptation with a single adaptation step and four iterations vs scratch model training. 12MFCC+12Δ MFCC were used as features.Top Left:30 sec of Training;Top Right:1 min of Training;Bottom Left:2 min Training;Bottom Right:4.25 min of Training of training.

As indicated at the beginning of Section 8.2, target models constructed by adapting speaker background models using Bayesian adaptation techniques can be advantageous when relatively small amounts of training data are available. To obtain good performance the background models should be constructed from large amounts of training

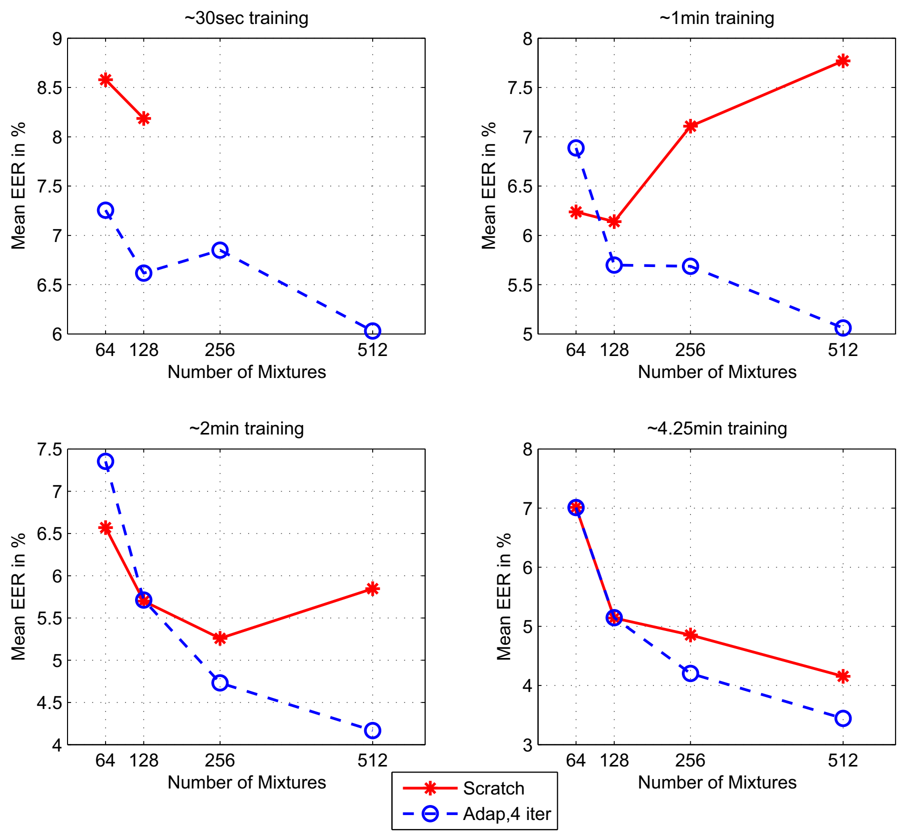


Figure 8.12: A comparison of the performance of Bayesian Adaptation with four iterations vs scratch model training. 12MFCC+12Δ+12$\Delta\Delta$ MFCC were used as features.Top Left:30 sec of Training;Top Right:1 min of Training;Bottom Left:2 min Training;Bottom Right:4.25 min of Training of training.

Training dur(mins)	no. of components			
	64	128	256	512
All	5.6/2	5.1/0.6	4.6/0.6	4.2/0.4
2	6.6/3.4	5.9/2.0	5.3/1.2	6.3/0.8
1	6.6/2.8	6.8/5.0	7.4/5.0	7.6/4.8
0.5	8.3/5.0	9.0/5.0	9.8/5.0	-

Table 8.4: Average and median (avg/mdn) equal-error rates in% as a function of training data and no. of mixture components for scratch models. No energy trimming is performed and the features used are 12MFCC+12ΔMFCC.

Training dur(mins)	no. of components			
	64	128	256	512
All	6.0/4.6	4.6/2.8	4.6/3.2	4.3/1.8
2	6.6/3.8	5.1/2.4	4.8/5.0	4.9/0.8
1	5.9/4.2	5.5/4.0	5.7/5.0	5.6/5.6
0.5	6.7/5.0	5.7/5.0	6.2/5.2	6.9/5.0

Table 8.5: Average and median (avg/mdn) equal-error rates in% as a function of training data and no. of mixture components for adapted models. Four iterations of adaptation were employed. No energy trimming is performed and the features used are 12MFCC+12ΔMFCC.

data from many speakers. The number of mixture components should also be large to give adaptation a chance to capture features closely matching the target speaker. To illustrate the effects on performance of adapted models with reduced training data we have selected the -100dB trimming level that results in no vectors omitted from either the test or training data.

The results are presented with the first feature set consisting of 12MFCC + 12ΔMFCC features and the second feature set consisting of 12MFCC + 12ΔMFCC + 12ΔΔMFCC features. Reference results for models created from scratch are shown in Table 8.4 for the first feature set and in Table 8.6 for the second feature set. Results for models created by Bayesian adaptation of speaker background models, in which the adaptation is iterated 4 times(there are indications that slightly better performance is obtained with adaptation iterated 4 times as compared with a single adaptation step), are shown in Table 8.5 for the first feature set and in Table 8.7 for the second feature set. Results are shown as a function of the amount of training data, ranging from all the training data (approx. 4.25 mins. average), 2 min.,1 min. and 0.5 min. of training data, and the

Training dur(mins)	no. of components			
	64	128	256	512
All	7.0/4.4	5.1/2.2	4.8/0.6	4.1/0.4
2	6.6/5.0	5.7/1.8	5.26/0.8	5.85/0.6
1	6.2/3.3	6.1/3.9	7.1/1.6	7.8/4.1
0.5	8.6/5.0	8.2/5.0	-	-

Table 8.6: Average and median (avg/mdn) equal-error rates in% as a function of training data and no. of mixture components for scratch models. No energy trimming is performed and the features used are 12MFCC+12ΔMFCC+12ΔΔMFCC.

Training dur(mins)	no. of components			
	64	128	256	512
All	7.0/5.0	5.1/2.0	4.2/1.8	3.4/0.8
2	7.4/5.0	5.7/2.9	4.7/1.6	4.17/1.0
1	6.9/5.0	5.7/4.5	5.7/4.0	5.1/2.3
0.5	7.3/8.1	6.6/5.0	6.9/5.0	6.0/3.8

Table 8.7: Average and median (avg/mdn) equal-error rates in% as a function of training data and no. of mixture components for adapted models. Four iterations of adaptation were employed. No energy trimming is performed and the features used are 12MFCC+12ΔMFCC+12ΔΔMFCC.

number of mixture components, ranging from 64 to 512. Fig. 8.11(for the first feature set) and Fig. 8.12(for the second feature set) combine these results into four different plots(one each for the amount of training data) that are displayed together. For the purpose of comparison, in Fig. 8.11, the mean equal-error rates for the adaptation of the speaker models with a single adaptation step are also presented.

For the reference results the general trend, as expected, is that performance degrades as the amount of training data decreases. In addition, when all the training data is used performance increases as the number of mixture components increases up to 512 components. However, as the training data is truncated performance degrades for models with large numbers of mixture components. This is expected since smaller amounts of training data cannot adequately support large models. In the most extreme conditions, namely 0.5 mins. training data and 512 components for the first feature set, and 0.5 mins. training data and 256 and 512 components for the first feature set, no models can be constructed because the amount of training data per component falls below the allowable threshold set for the computation.

For the results for models created by adaptation of background models, the same general degradation in performance is seen as the amount of training data decreases. However, the degradation is significantly less than the case with no adaptation. In addition, for the most part, performance actually improves as the model size increases, even for smaller amounts of training data. The contrast in performance between scratch and adapted models can be seen clearly for both feature sets from the plots in Figs. 8.11 and 8.12. As an example for the first feature set, for 2 mins. of training data and 512 components the scratch model equal-error rate was 6.3% while the adapted model performance was 4.9%. From Fig. 8.11 it can also generally be inferred that slightly better performance is obtained for adapted models where the adaptation was carried out with 4 iterations. For both feature sets, even when using all the training data available, models created by adaptation of background models performed generally as well or better than scratch models. An additional observation that can be made for scratch versus adapted model performance is that the median equal-error rate is generally higher for adapted models compared with scratch models. This indicates a more uniform performance across speakers for adapted models.

It is also important to determine which among the two feature sets actually gave better performance. For this it is instructive to look at the plot in Fig. 8.13, which collectively plots the average equal-error rates for target models obtained by adaptation, by iterating four times. Between the two it is clear that the performance of the 36 dimensional features consisting of 12MFCC + 12ΔMFCC + 12ΔΔMFCC is superior. Even though a degradation in the performance is observable for this feature set as the amount of training data is reduced, it is clear that the degradation in this case is much lower. For example for 0.5 min of training, adapted models with 24-dimensional features had a mean equal-error rate of 6.9% compared to 6% in the case of 36 dimensional features. In fact, the lowest equal-error rates, for different amounts of data, were obtained using the 36-dimensional feature set with adapted models with 512 mixtures. The lowest equal-error rate 3.4% was obtained with this configuration, while using all available training data. The individual equal-error rates for this configuration are shown in Table 8.8.

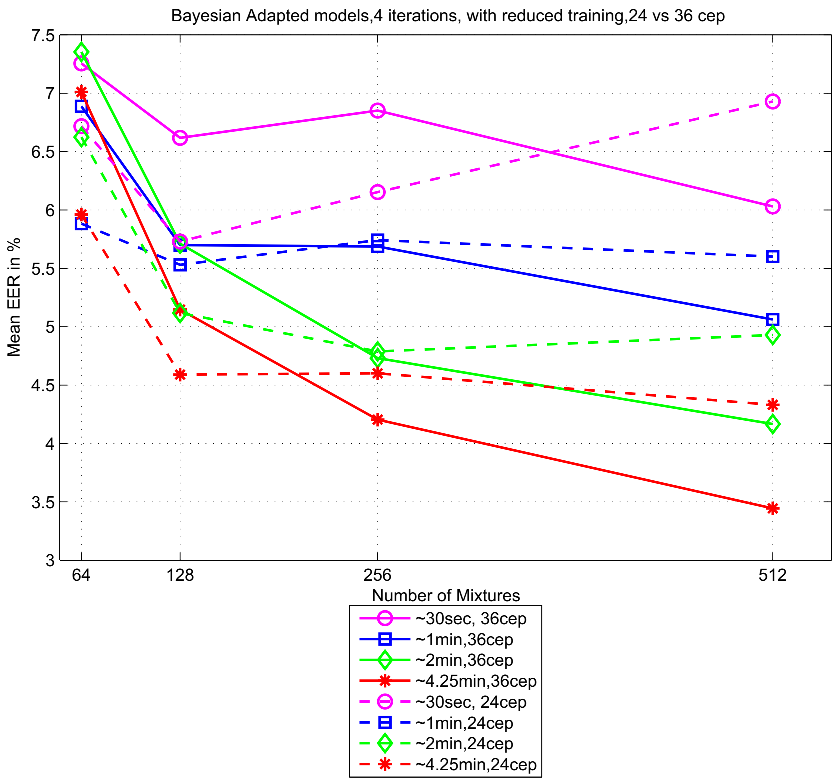


Figure 8.13: A comparison between the use of 24 and 36 features, while employing Bayesian Adaptation with four iterations.

Spkr	EER(%)
WS1	0.0
CM1	0.6
JW1	0.0
JM1	10.4
JS1	0.8
IS1	3.5
JW2	0.2
KM1	0.2
AM2	5.0
AG1	0.4
MA1	5.6
NR1	1.9
LR1	0.0
AR1	0.0
OH1	10.4
JD1	4.4
DG1	15.0
avg/mdn	3.4/0.8

Table 8.8: Equal-error rates in % for 17 speakers, for adapted models with 512 mixtures, trained using 4.25 min. of training data. The features used were 12MFCC + 12ΔMFCC +12ΔΔMFCC.

8.2.4 Experiments with longer testing utterances and two-session training

Average and median equal-error rates as a function of test trial length and number of mixture components for both target and background speaker models are shown in Table 8.9. Test trial length refers to the number of test utterances used in a single test trial. Recall that there were 20 test utterances available for each speaker. When the test trial length was set to 1 there were 20 test trials for each speaker each comprising 1 test utterance. When the test trial length was set to a number greater than 1, successive test utterances were combined to form a test trial. Thus when the test trial length was set to 2, the first test trial comprised test utterances 1 and 2, the second trial, test utterances 3 and 4, and so forth. When the test trial length was 2 there were 10 test trials per speaker and when the test trial length was 3 there were 6 test trials per speaker. Because the number of test trials for the true speaker was reduced significantly when the test trial length was increased, a control experiment was carried out in which the number of test trials remained at 20 by overlapping the test utterances for each trial. Thus, for test trial length set to 2, the first test trial consisted of test utterances 1 and 2, the second, 2 and 3, the third 3 and 4, and so forth. In this case the test trials were no longer independent. However, the results for overlapped test trials were essentially the same as non-overlapped test trials.

The results shown in the table indicate that performance improves both as the number of mixture components and the test trial length increase. However, performance improvements are greater as a function of test trial length. The best performance obtained, for test trial length equal to 3 and 128 mixture components averaged 3.2% equal-error rate with a median equal-error rate performance of 0. Although the performance has improved significantly over the baseline performance, there is still a large discrepancy between the average and median error rates.

Individual equal-error rates for this combination of conditions are shown in Table 8.10. Comparing the performance in this table with the individual performances shown in Table 6.1 it can be seen that there are significant improvements for almost all speakers. 11 speakers now have equal-error rate equal to 0. However, the error rate for speaker DG1 has improved only from 40.0% to 33.3%. It turned out that there was marked difference in sound quality between the two recording sessions for this speaker. Since the training utterances were from session 1 and the test utterances were from session 2, it follows that there was a significant mismatch between the reference and test conditions for this speaker. Because of the mismatch, test trials for this speaker would be repeatedly rejected. In this situation the customer could request retraining to produce new models. The data obtained from the new training session could then be combined with the original training data to provide a new model. This newly trained model should incorporate the mismatched conditions and thereby be tolerant of test utterances matched to either of the conditions. Although we did not have sufficient data from target speakers to simulate this situation, we could provide a flawed simulation by retraining the speaker model incorporating the data from both recording sessions and retesting. The flaw is that the testing data is no longer independent of the training data since it comes from the same session as half of the new training data.

test length	no. of components		
	32	64	128
1 utt.	7.4/1.6	6.3/1.8	5.2/1.0
2 utt.	5.8/0.8	3.7/0.4	3.9/0.0
3 utt.	3.7/0.0	3.5/0.0	3.2/0.0

Table 8.9: Average and median (avg/mdn) equal-error rates in % as a function of no. of mixture components and test trial length

Spkr	EER(%)
WS1	0.0
CM1	0.6
JW1	0.0
JM1	16.7
JS1	0.0
IS1	1.3
JW2	0.0
KM1	0.0
AM2	0.0
AG1	0.0
MA1	0.0
NR1	1.3
LR1	0.0
AR1	0.0
OH1	0.0
JD1	2.0
DG1	33.3
avg/mdn	3.2/0.0

Table 8.10: Individual Equal-error rates for 17 speakers with 3 utterances per test trial, 128 components GMMs and -35dB energy trimming level

test length	no. of components		
	32	64	128
1 utt.	3.4/0.6	3.2/0.4	2.2/0.4
2 utt.	2.0/0.0	1.1/0.0	0.6/0.0
3 utt.	0.8/0.0	0.4/0.0	0.2/0.0

Table 8.11: Average and median (avg/mdn) equal-error rates in % as a function of no. of mixture components and test trial length with 2-session training

The results for such 2-session training are shown in Table 8.11. It can be seen that these results are greatly improved. The best result, for 3-utterance test trials and 128 components was just 0.2% average equal-error rate and 0.0% median. These results are only suggestive, however, because of the above-mentioned flaw and it should also be kept in mind that all the target speakers here have retrained, not just the markedly mismatched speaker DG1. A more representative condition is to substitute the 2-session performance only for DG1. With 2-session training this speaker's error rate falls to 0 and the revised performance for 3-utterance test trials and 128 component models becomes 1.2% average equal-error rate and 0.0% median equal-error rate.

8.2.5 Experiments with variance limiting

Training	no. of components			
dur(mins)	64	128	256	512
All	5.6/2	5.1/0.6	4.6/0.6	4.2/0.4
2	6.6/3.4	5.9/2.0	5.3/1.2	6.3/0.8
1	6.6/2.8	6.8/5.0	7.4/5.0	7.6/4.8
0.5	8.3/5.0	9.0/5.0	9.8/5.0	-

Table 8.12: Average and median (avg/mdn) equal-error rates in % for scratch models with no variance limiting applied to the nodal variances

Training	no. of components			
dur(mins)	64	128	256	512
All	5.75/2.2	5.27/1.2	4.4/0.6	4.5/0.6
2	5.5/2.8	5.9/2.4	5.5/1.0	5.9/2.6
1	6.0/2.4	6.9/5.0	7.3/3.4	7.7/5.0
0.5	8.3/5.0	9.6/5.0	9.6/5.0	-

Table 8.13: Average and median (avg/mdn) equal-error rates in % for scratch models with variance limiting applied to the nodal variances.

When there is insufficient data to reliably estimate the values for the variances of the Gaussian mixture components can become quite small in magnitude. These small variances produce a singularity in the model's likelihood function and could degrade verification performance.

In order to avoid these singularities, a variance limiting constraint was applied to each variance value along each dimension of the $D \times D$ covariance matrix(recall that

D is the number of features in each feature vector), for each component mixture of the Gaussian density. Therefore for a particular Gaussian the mixture, the variance value $\sigma_i^2$for $i = 1, \ldots, D$ the variance limiting constraint is applied as,

$$\sigma_i^2 = \begin{cases} \sigma_i^2 & \text{if } \sigma_i^2 > m_{\sigma_i^2} - v_{\sigma_i^2} \\ m_{\sigma_i^2} - v_{\sigma_i^2} & \text{if } \sigma_i^2 < m_{\sigma_i^2} - v_{\sigma_i^2} \end{cases}$$

where was $m_{\sigma_i^2}$ the estimate of average value of σ_i^2 and $v_{\sigma_i^2}$ was an estimate of the variance of σ_i^2 in the ith dimension, calculated over all mixture components. The limiting therefore seeks clips the variance at a value one standard deviation away from the mean variance value. This limiting was applied after each iteration of the EM procedure.

Tables 8.12 and 8.13 show results for the cases where scratch model training was carried out with and without variance limiting. The results indicated that applying variance limiting does not really help nor hurt. One could draw the conclusion that for majority of mixture components the values of the variances are closer to or greater than the estimated average value of the variance, and do not become small enough for variance limiting to have a major effect. In other words there seems to be enough data to reliably estimate the variance components.

8.3 Conclusions

The following conclusions can be drawn from the experimental results. Models created by adaptation of background models perform as well or better than scratch models. The advantage is especially significant when only smaller amounts of training data are available. Additionally, the 36-dimensional feature set with 12MFCC + 12ΔMFCC + 12$\Delta\Delta$MFCC seems to be the best configuration of features to use along with Bayesian Adaptation, while employing larger size models($\geq$ 512 mixtures). With respect to test trial length, it was found that performance for test trials consisting of a single 5- to 7-word sentence utterance was associated with inadequate performance. Performance improved significantly with 2-sentence test trials and continued to improve with 3-sentence test trials. Mismatched training and test conditions can lead to markedly poor performance. It was shown that it was possible to remediate this condition if it was detected by using an additional training session with additional data to retrain the models. With respect to the amount of training data used to train models, it was shown that performance degraded fairly steadily as the amount of training data was reduced. It is best to enroll and train new speaker models with the greatest practicable amount of training data.

References

[1] Institute for Signal & Information Processing at Mississippi State University. Fundamentals of speech recognition:a tutorial based on a public domain c++ toolkit, January 2002. http://www.ece.msstate.edu/research/isip/projects/speech/.

[2] Fred Cummins. Sound patterns in human language: The vocal tract, 1998. http://cspeech.ucd.ie/∼fred/teaching/oldcourses/phonetics/vtract.html.

[3] Don Johnson. Modeling the speech signal, June 2008. http://cnx.org/content/m0049/latest/.

[4] D. A. Reynolds, Thomas F. Quatieri, and Robert B. Dunn. Speaker verification using adapted gaussian mixture models. *Digital Signal Processing*, 10:19–41, January.

[5] D. A. Reynolds. An overview of automatic speaker recognition technology. In *Acoustics, Speech, and Signal Processing, 2002. Proceedings. (ICASSP '02). IEEE International Conference on*, volume 4, pages 4072–4075, May 2002.

[6] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *Acoustics, Speech, and Signal Processing [see also IEEE Transactions on Signal Processing], IEEE Transactions on*, 28(4):357–366, August 1980.

[7] Douglas A. Reynolds. Speaker identification and verification using gaussian mixture speaker models. *Speech Commun.*, 17(1-2):91–108, 1995.

[8] D. A. Reynolds. *A Gaussian Mixture Modeling Approach to Text-Independent Speaker Identification*. PhD thesis, Georgia Institute of Technology, Atlanta, GA, USA, August 1992.

[9] D. A. Reynolds and R. C. Rose. Robust text-independent speaker identification using gaussian mixture speaker models. *IEEE Transactions on Speech and Audio Processing*, 3(1):72–83, January 1995.

[10] L. R. Rabiner and R. W. Schafer. *Digital Processing of Speech Signals*. Prentice Hall, Upper Saddle River, New Jersey, USA, 1978.

[11] A.E. Rosenberg. *Springer Handbook of Speech Processing*, chapter Overview of Speaker Recognition. Springer New York, 2008.

[12] A. E. Rosenberg. Automatic speaker verification: A review. *Proceedings of the IEEE*, 64(4):475–487, April 1976.

[13] A. L. Higgins, L. G. Bahler, and J. E. Porter. Voice identification using nearest-neighbor distance measure. In *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*, volume 2, pages 375–378, Minneapolis, MN, USA, April 1993.

[14] J. Oglesby and J. S. Mason. Radial basis function networks for speaker recognition. In *Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on*, pages 393–396, Toronto, Ont., Canada, April 1991.

[15] K. R. Farrell, R. J. Mammone, and K. T. Assaleh. Speaker recognition using neural networks and conventionalclassifiers. *IEEE Transactions on Speech and Audio Processing*, 2:194–205, January 1994.

[16] J. Oglesby and J. S. Mason. Optimisation of neural models for speaker identification. In *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on*, pages 261–264, Albuquerque, NM, USA, April 1990.

[17] V. Wan and W. M. Campbell. Support vector machines for speaker verification and identification. In *Neural Networks for Signal Processing X, 2000. Proceedings of the 2000 IEEE Signal Processing Society Workshop*, volume 2, pages 775–784, Sydney, NSW, Australia, 2000.

[18] M. Schmidt and H. Gish. Speaker identification via support vector classifiers. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, volume 1, pages 105–108, Atlanta, GA, USA, May 1996.

[19] Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *Communications, IEEE Transactions on [legacy, pre - 1988]*, 28(1):84–95, January 1980.

[20] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, November 2000.

[21] F. Soong, A. Rosenberg, L. Rabiner, and B. Juang. A vector quantization approach to speaker recognition. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '85.*, volume 10, pages 387–390, April 1985.

[22] H. Gish, K. Karnofsky, M. Krasner, S. Roucos, R. Schwartz, and J. Wolf. Investigation of text-independent speaker indentification over telephone channels. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '85.*, volume 10, pages 379–382, April 1985.

[23] A. B. Poritz. Hidden markov models: a guided tour. In *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on*, pages 7–13, New York, NY, USA, April 1988.

[24] L. Rabiner and B. Juang. An introduction to hidden markov models. *IEEE [see also IEEE Signal Processing Magazine] ASSP Magazine*, 3:4–16, January 1986.

[25] A. E. Rosenberg, C. H. Lee, and F. K. Soong. Sub-word unit talker verification using hidden markov models. In *Acoustics, Speech, and Signal Processing, 1990.*

ICASSP-90., 1990 International Conference on, pages 269–272, Albuquerque, NM, USA, April 1990.

[26] T. Matsui and S. Furui. Comparison of text-independent speaker recognition methods usingVQ-distortion and discrete/continuous HMM's. *IEEE Transactions on Speech and Audio Processing*, 2(3):456–459, July 1994.

[27] A. E. Rosenberg and S. Parthasarathy. Speaker background models for connected digit password speakerverification. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, volume 1, pages 81–84, Atlanta, GA, USA, May 1996.

[28] S. Parthasarathy and A. E. Rosenberg. General phrase speaker verification using sub-word backgroundmodels and likelihood-ratio scoring. In *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*, volume 4, pages 2403–2406, Philadelphia, PA, USA, October 1996.

[29] F. Bimbot, J. F. Bonastre, C. Fredouille, G. Gravier, Magrin I. Chagnolleau, S. Meignier, T. Merlin, Ortega J. Garcia, Petrovska Delacretaz, and Reynolds. A tutorial on text-independent speaker verification. *EURASIP Journal on Applied Signal Processing*, 4:430–451, 2004.

[30] G. R. Doddington. Speaker recognitionidentifying people by their voices. *Proceedings of the IEEE*, 73(11):1651–1664, November 1985.

[31] M. Slaney. Auditory toolbox version 2, technical report 1998-10, interval research corporation. http://cobweb.ecn.purdue.edu/ malcolm/interval/1998-010/.

[32] J. L. Gauvain and Chin-Hui Lee. Maximum a posteriori estimation for multivariate gaussian mixtureobservations of markov chains. *IEEE Transactions on Speech and Audio Processing*, 2(2):291–298, April 1994.

[33] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38, 1977.

[34] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, August 2006.

[35] Leonard E. Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.

[36] A. Papoulis and S. Unnikrishna Pillai. *Probability, Random Variables, and Stochastic Processes*. Mc-Graw Hill, 2002.

[37] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, San Diego, CA, USA, 1972.

[38] Kai-Fu Lee. *Automatic Speech Recognition : The Development of the SPHINX System*. Kluwer Academic Publishers, 1989.

[39] P. Clarkson and R. Rosenfeld. Statistical language modeling using the cmu-cambridge toolkit. In *Proc. Eurospeech*, 1997.

[40] C. Bennett and A.I. Rudnicky. The carnegie mellon communicator corpus. In *Proceedings of ICSLP 2002 (Denver, Colorado)*, pages 341–344, 2002.

[41] Donald E. Knuth. *The Art of Computer Computing*, volume 2, chapter 3. Addison-Wesley, 1998.

[42] Tom Christiansen and Nathan Torkington. The perl faq. http://perldoc.perl.org/perlfaq4.html.