ABSTRACT

Title of thesis: ROBUST VOICE MINING TECHNIQUES FOR TELEPHONE CONVERSATIONS

Sandeep Manocha, Master of Science, 2006

Thesis directed by: Dr. Carol Y. Espy-Wilson
Department of Electrical Engineering

Voice mining involves speaker detection in a set of multi-speaker files. In published work, training data is used for constructing target speaker models. In this study, a new voice mining scenario was considered, where there is no demarcation between training and testing data and prior target speaker models are absent. Given a database of telephone conversations, the task is to identify conversations having one or more speakers in common. Various approaches including semi-automatic and fully automatic techniques were explored and different scoring strategies were considered. Given the poor audio quality, automatic speaker segmentation is not very effective. A new technique was developed which does not require speaker segmentation by training a multi-speaker model on the entire conversation. This technique is more robust and it outperforms the automatic speaker segmentation approach. On the ENRON database, the EER is 15.98% and 6.25% for at least one and two speakers in common, respectively.

ROBUST VOICE MINING TECHNIQUES FOR TELEPHONE CONVERSATIONS

by

Sandeep Manocha

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2006

Advisory Committee:
Dr. Carol Y. Espy-Wilson, Chair/Advisor
Dr. Shihab Shamma
Dr. Min Wu

ACKNOWLEDGMENTS

First of all, I would like thank to my mother, father and sister. They have been very supportive and have always motivated me.

I am very grateful to my advisor, Prof. Carol Espy-Wilson. It has been a great experience working with her and I would like to thank her for all the knowledge and advice that she has given. She has been very understanding and I would like to thank her for all the encouragement, support and guidance.

I would like to thank Prof. Shihab Shamma and Prof. Min Wu for agreeing to serve on my thesis committee and for sparing their invaluable time reviewing the manuscript.

Special thanks to all the members of the lab: Om, Tarun, Srikanth, Xinhui, Daniel and Vikramjit. Om and Tarun have been very supportive and have helped me a lot. Srikanth has been a great friend and has helped me out on numerous occasions. Thanks to Daniel for his valuable suggestions and help in the project. I would like to thank Olakunle for doing manual segmentation of the telephone conversations.

I would like to thank Dr. Douglas Reynolds for the MIT-LL GMM System and for his prompt responses to the queries that I had.

I would like to express my gratitude to Peggy, Carlos, Cliff and Luther for their technical support and help with the cluster. Thanks to the cluster for all the

data crunching !!

I would also like to thank my roommates Deepak, Eswaran, Karthik and Rajesh. Thanks for being such wonderful roommates and making me watch Tamil movies !!

I would like to thank all my friends here at the University of Maryland. Thanks to Indrajit, Kaushik, Anniruddha, Sameer, Pavan, Mahesh, Ashwin, Anuj, Abhinav, Kaustav, Soma, Supratik, Ayush, Subhamoy and Anyesha for their help and support. Special thanks to Indrajit, Kaushik and Mahesh for their company in A.V. Williams on weekends !!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

The speech signal conveys various levels of information. Primarily, the speech signal conveys the message with the words of the language. There are other levels of information such as the gender and identity of the speaker, the speaker's emotion and the language being spoken. The area of speech recognition is concerned with the message conveyed by the speech signal, while the area of speaker recognition aims to extract the identity of the speaker from the speech signal. The field of speaker recognition has numerous practical applications such as security systems (that provide control of physical entry and information access), transaction authorization in telephone banking, forensics, voice mining of speech databases, speech data management (voice mail browsing or intelligent answering machines), etc. [1].

1.1 Speaker Recognition

Speaker recognition is a general term used to include the different tasks of discriminating people based on their voice. There are many terms that have been used in the literature to distinguish different tasks, including speaker identification, speaker verification, speaker detection, speaker spotting, speaker tracking and voice mining. Speaker recognition can be mainly divided into two fundamental tasks: speaker identification and speaker verification [2].

Speaker identification is the task of determining who is talking from a set of known voices or speakers. The unknown person makes no identity claim and so the system has to select one speaker from the population of N speakers. Generally it is assumed that the unknown voice must come from a fixed set of known speakers and it is a closed set problem. Speaker verification is the task of verifying a person's claimed identity and the system has to take a binary (yes/no) decision. It is treated as an open set problem since usually the imposters (those falsely claiming to be a valid user) are not known to the system.

The tasks can further be subdivided as text-dependent and text-independent depending on the level of user cooperation and control in an application. In a text-dependent application, the system has prior knowledge of the text to be spoken and it is expected that the user will cooperatively speak this text. In a text-independent application, there is no prior knowledge of the text to be spoken. The performance of text-dependent systems is superior compared to text-independent systems; however the text-independent systems offer greater flexibility. As the speech recognition systems are merged with speaker recognition systems, the recognition accuracy improves and the distinction between text-dependent and text-independent systems diminishes.

The essential steps involved in the speaker recognition task are: (1) digital speech data acquisition (2) feature extraction (3) speaker modeling (4) pattern matching and decision making. However, we will assume that the speech data is already available and we will not be dealing with the first step. The remaining three steps will be discussed in this chapter and previous work in these areas has also been

outlined.

1.2 Feature Extraction

The goal of feature extraction is to extract speaker-specific features from the speech signal that will enable to discriminate between the speakers. Speaker-related differences are as a result of a combination of anatomical differences inherent in the vocal tract and learned speaking habits of different individuals. The features attempt to capture these differences that are manifested in the speech signal. The features can be divided into two categories: low-level features and high-level features.

The low-level features based on purely acoustic signals are computed over small intervals of speech (10 - 20 ms) across the entire utterance yielding a feature vector every 10 ms. These features are computed at a high frame rate since speech is a quasi-stationary signal and they aim to capture the instantaneous properties of the signal. The high-level features are computed over larger amount of speech (can be the entire utterance) and aim to capture learned characteristics of the speaker such as the speaking rate, prosodic effects and dialect [3]. The high-level information can add complementary knowledge to the low-level features and are possibly more robust to acoustic degradations from channel and noise effects, to which low-level features are highly susceptible. However, the disadvantage of these high-level features is that they need relatively larger amount of speech data to reliably compute these features. In this study, we have only used low-level features for speaker recognition and whenever the terms features or parameters are used, it refers to the low-level

features.

Various features have been developed in the past for speaker recognition such as linear prediction coefficients (LPC), cepstral coefficients, mel-frequency cepstral coefficients (MFCCs), etc. The LP coefficients typically are nonlinearly transformed into perceptually meaningful domains suited for the application. Some useful feature domains include reflection coefficients (RCs); log-area ratios (LARs) or arcsin of the RCs; LSP frequencies and the LP cepstrum [4]. The mel-frequency cepstral coefficients (MFCCs),which do not require LP analysis are the most popular among these coefficients and they have been widely used for both speaker and speech recognition systems.

Traditional speaker recognition systems rely on the vocal tract dynamics to discriminate between the speakers and under-emphasize the significance of the source. However, more recent work has shown that the addition of source information can prove to be valuable speaker-specific information [5]. The MFCCs implicitly code the vocal tract information and some source information in them. Acoustic Parameters (APs) have been developed that attempt to explicitly capture the source information and different vocal tract configurations. These eight parameters have a better performance for female speakers than that of the 26 MFCCs and 39 MFCCs and on the average they have comparable performance for varying population size [6].

1.3 Speaker Modeling

Speaker models are constructed using the features that are extracted from the speech signal. The pattern matching algorithm compares the features of the test signal with the speaker models. There are two types of speaker models: stochastic models and template models. For stochastic models, the pattern matching is probabilistic and it computes a likelihood of the observation given a speaker model. In the case of template models, the pattern matching is deterministic. The observation is assumed to be an imperfect replica of the template and a distance measure is computed between the observation vectors and the template model. Examples of template-based models are Dynamic Time Warping (DTW), Vector Quantization Source Modeling and Nearest Neighbors (NN) [4]. Hidden Markov Models (HMMs) [7] are an example of stochastic models and single state HMMs also known as Gaussian Mixture Models (GMMs) [8] are popularly used. The HMMs are used for text-dependent applications where the phrases or phonemes are modeled using the multi-state left-to-right structure. The GMMs which do not use such temporal information and attempt to form a statistical representation of the speaker are used for text-independent speaker recognition. The template models dominated early work in text-dependent speaker recognition; however the stochastic models are now state-of-the-art. These models efficiently model statistical variation of the features and yield superior recognition performance compared to the template-based models. Since in this study, the focus is on text-independent speaker recognition, the GMMs have been used to model the speakers.

1.3.1 Gaussian Mixture Models (GMMs)

A Gaussian mixture model consists of a weighted sum of M component Gaussian densities and is given by the equation:

$$p(\vec{x}|\lambda) = \sum_{i=1}^{M} p_i b_i(\vec{x}) \tag{1.1}$$

where $\vec{x}$ is a D-dimensional random vector, $b_i(\vec{x})$, i=1,...,M are the component densities and p_i, i=1,...,M are the mixture weights. Each component density is a D-variate Gaussian function of the form:

$$b_i(\vec{x}) = \frac{1}{(2\pi)^{D/2}\Sigma_i^{1/2}} exp\left\{ -\frac{1}{2}(\vec{x} - \vec{\mu_i})'\Sigma_i^{-1}(\vec{x} - \vec{\mu_i}) \right\} \tag{1.2}$$

with mean vector $\vec{\mu_i}$ and covariance matrix Σ_i. The mixture weights satisfy the following constraint:

$$\sum_{i=1}^{M} p_i = 1 \tag{1.3}$$

The complete Gaussian mixture density is parameterized by the mean vectors, covariance matrices and mixture weights from all component densities. The collection of parameters that make up the speaker model (λ) can be represented as:

$$\lambda = \{p_i, \vec{\mu_i}, \Sigma_i\} \quad i = 1, ..., M \tag{1.4}$$

The GMM can have different forms depending on the choice of covariance matrices. The three types of covariance matrices are:

1. Nodal covariance: one covariance matrix per Gaussian component
2. Grand covariance: one covariance matrix for all Gaussian components in a speaker model

3. Global covariance: a single covariance matrix shared by all speaker models

The covariance matrix can also be full or diagonal. It has been empirically observed that nodal, diagonal covariance matrices yield better speaker recognition accuracies [8].

There are two principal motivations for using GMMs to construct a speaker model [8]. Firstly, the individual component densities of the multi-modal density can model the set of underlying acoustic classes such as vowels, fricatives and nasals. The properties of each acoustic class are represented by the Gaussian density of the class. The second motivation for using Gaussian mixture densities is the empirical observation that a linear combination of Gaussian basis functions is capable of representing a large class of sample distributions. GMMs have the ability to form smooth approximations to arbitrarily-shaped densities.

Speaker models are constructed by estimating the parameters of the GMM from the available training data of the speaker. The most popular and well-established method for training is maximum likelihood (ML) estimation. ML estimation seeks to find the model parameters that maximize the likelihood of the GMM given the training data. For a sequence of T training vectors X = $\{\vec{x_1}, ..., \vec{x_T}\}$ and using the assumption that the feature vectors are independent, the GMM likelihood can be written as:

$$p(X|\lambda) = \prod_{t=1}^{T} p(\vec{x_t}|\lambda) \tag{1.5}$$

Since this expression is a nonlinear function of the parameters,direct maximization is not possible. The ML parameter estimates are obtained iteratively using the

expectation-maximization (EM) algorithm. In each iteration, a new model ($\bar{\lambda}$) is estimated from the previous model (λ) such that:

$$p(X|\bar{\lambda}) \geq p(X|\lambda) \tag{1.6}$$

The new model then becomes the initial model for the next iteration and the process is repeated until some convergence threshold is reached.

1.4 Pattern Matching and Decision Making

For pattern matching, the features of the test utterance are scored against the speaker model. Since we are using stochastic models, the computed score is the probability of the sequence of feature vectors given the speaker model. Since we assume that the feature vectors are independent, the probability is given by equation 1.5, except that the feature vectors are now from the test utterance rather than the training data. The equation can also be expressed as the sum of log probabilities given by:

$$p(X|\lambda) = \sum_{t=1}^{T} log\ p(\vec{x_t}|\lambda) \tag{1.7}$$

The decision making process depends on the speaker recognition task. In this section, the decision rules for the two primary tasks: speaker identification and speaker verification are discussed.

1.4.1 Speaker Identification

In speaker identification, the system has to decide who among the candidate speakers said the utterance. This is an N-class decision task, where N is the number

of candidate speakers or also known as the size of the population. As noted earlier, speaker identification is usually treated as a closed set problem, where the actual speaker is always one of the candidate speakers. Hence the objective is to find the speaker model which has the maximum a posteriori probability for a given observation sequence. This can expressed by the following equation:

$$\hat{S} = arg \max_{1 \leq k \leq N} p(\lambda_k | X) \tag{1.8}$$

where $\hat{S}$ represents the selected speaker. Using Bayes' rule, the probability can be converted to the following form:

$$\hat{S} = arg \max_{1 \leq k \leq N} \frac{p(X|\lambda_k) Pr(\lambda_k)}{p(X)} \tag{1.9}$$

Assuming that the speakers are equally likely, i.e. $Pr(\lambda_k) = \frac{1}{N}$ and noting that $p(X)$ is the same for all speaker models, the classification rule simplifies to:

$$\hat{S} = arg \max_{1 \leq k \leq N} p(X|\lambda_k) \tag{1.10}$$

Using equation 1.7, we can give the decision rule as:

$$\hat{S} = arg \max_{1 \leq k \leq N} \sum_{t=1}^{T} log\ p(\vec{x_t}|\lambda_k) \tag{1.11}$$

The identification error rate (EID) is used as a performance measure and is given by:

$$EID = \frac{n_{err}}{n_{tot}} \tag{1.12}$$

where n_{tot} and n_{err} are the total number of trials and the number of trials in error, respectively. The chance accuracy of the system is 1/N and the error rates usually increase as the population size increases.

1.4.2 Speaker Verification

Speaker verification is the task of deciding, given a sample of speech, whether a specified candidate speaker said it. This is a two-class decision task and since it fits into detection theory, it is also referred to as a speaker detection task. The two classes for verification are: the specified speaker (known as the "target speaker") and some speaker other than the specified speaker (known as "impostor" or "non-target speaker"). Usually, it is assumed that the verification system does not have knowledge of who the impostor speakers are and it is treated as an open set problem. An impostor model is widely used in speaker verification and it can be crucial to obtaining good performance. An impostor model acts as a normalization to help reduce non-speaker related variability (eg. text spoken, microphone, noise) and the score of the impostor model is compared to the speaker model score.

Speaker verification is essentially a hypothesis testing problem. Let H_0 be the hypothesis that the speaker is an impostor and H_1 be the hypothesis that the speaker is indeed the target speaker. The speaker verification system essentially implements a likelihood ratio test to distinguish between the two hypothesis. Assuming equal costs of misclassification, the likelihood ratio test is given as:

$$L = \frac{p(X|H_1)}{p(X|H_0)} \tag{1.13}$$

if $L \geq \eta$ then choose H_1, else choose H_0.

The threshold can be determined by using an estimate of the prior probabilities of the target speaker and the impostor or to achieve a fixed value of false acceptance or false rejection. The performance of the verification system can be characterized

in terms of two error measures: the miss error rate (or false rejection rate) and false alarm error rate (or false acceptance error rate). These correspond respectively to the probability of not detecting the target speaker when present and the probability of falsely detecting the target speaker when not present. The error measures are computed as:

$$E_{miss} = \frac{n_{miss}}{n_{target}} \tag{1.14}$$

$$E_{fa} = \frac{n_{fa}}{n_{impostor}} \tag{1.15}$$

where in the first equation n_{target} and n_{miss} are the number of target trials and the number of those where the target speaker was not detected, respectively. In the second equation, $n_{impostor}$ corresponds to the number of impostor trials and n_{fa} is the number of those where the target speaker was falsely detected. In order to obtain a single number performance figure, the equal error rate (EER) is commonly used. The threshold of the system is varied, till the two types of errors are equal and this value is the EER. The tradeoff between the miss and false alarm rates is a function of the decision threshold and it is depicted in the Receiver Operating Characteristic (ROC) curve. An improvement to this visualization aid was introduced by NIST [9] and it is known as Detection Error Trade-off (DET) plots. The miss and false alarm probabilities are plotted according to their corresponding Gaussian deviates, rather than the probabilities themselves. This results in a non-linear probability scale; however the trade-off curves usually appear as straight lines.

There are two dominant approaches used for imposter modeling in the likelihood ratio test [1]. The first approach, known as likelihood sets, cohorts or back-

ground sets, uses a collection of other speaker models to compute the imposter match score. The impostor match score is usually computed as a function, such as the maximum or average of the match scores from the set of models. The second approach, known as general, world or universal background modeling uses a single speaker-independent model trained on speech from a large number of speakers to represent speaker-independent speech. The advantage of this approach is that only a single cohort model needs to be trained and scored. The Maximum-A-Posteriori (MAP) training is widely used to adapt the background model to construct the speaker-specific model [10]. The Universal Background Model (UBM) and the MAP adaptation approach has been found to give superior performance in numerous speaker verification evaluations and has been used in this study [10].

1.5 Speaker Detection

In the previous section, the terms speaker verification and speaker detection have been used interchangeably and they were referring to the same task. However, these two terms have also been used [11] to distinguish between two different tasks. Speaker verification is the task of deciding, given a speech sample, whether it comes from the known target speaker, or not. It is assumed that the speech sample contains only single speaker data. When more than one speaker is present in the test sample (e.g. a two-sided conversation), the task is defined as speaker detection. In this report, we make this distinction between speaker verification and speaker detection and the above definitions are used. In the NIST Speaker Recognition evaluations

[12], [13], speaker verification is referred to as single speaker detection and speaker detection is known as multi-speaker speaker detection.

There are two primary approaches to the speaker detection problem. In one approach, speaker segmentation is used to turn the multi-speaker problem into a sequence of single speaker problems. This approach is referred to as an external segmentation [14] and the segmentation algorithm attempts to partition the speech file into speaker homogenous regions without any prior knowledge of the speakers. These segmented portions are scored against the target model as done in speaker verification. The other approach does not employ an explicit speaker segmentation algorithm to partition the test utterance. The target speaker model and the impostor model are used to compute the log-likelihood scores on a frame-by-frame basis. These scores are used to first partition the speech file into speaker homogenous regions and then to compute a score for these regions. This approach is referred to as internal segmentation [14]. A variation of this approach is to divide the multi-speaker file into short segments, which probably hold the voice of one speaker only [11]. As in speaker verification, log-likelihood scores are computed for each segment by comparing it to the target speaker model and the impostor model. A combination of these scores such as the maximum score or the average score above a fixed threshold can be used to obtain a score for the test file. This score can be used to make the speaker detection decision. Another new approach has recently been used for speaker detection that scores a test segment by comparing it directly to similar instances of that speech in training data [15]. This non-parametric technique achieved good results on the NIST 2001 Extended Data task [13], however it relies

on a large amount of training data for the target speakers (about 20 minutes of speech per speaker).

In most of the previous work on speaker detection, it is assumed that prior target models are available or single speaker training data for training the models is available. In the NIST 2002 speaker recognition evaluation [13], the two speaker detection task used two-speaker files for both enrollment and testing. However, the enrollment data for each target speaker is composed of three two-speaker files, where the target speaker is the only speaker that is common to all three files. Once again, training data for the target speaker could be created by using speaker segmentation and finding the common speaker present in the three files [16]. A speaker detection technique was developed without using prior speaker models to detect and model new speakers as they call into the system [17]. The target application for this system was an automated answering system, which can automatically identify and authenticate callers without the need for a speaker enrollment process. However, this speaker detection system was only applied and tested for single-speaker telephone transactions.

1.5.1 Voice Mining Task

Voice mining involves the detection of speakers in a set of multi-speaker files. In this study, a new voice mining scenario has been considered. This scenario was motivated by a problem encountered during the Joint Institute for Knowledge Discovery (JIKD) Email and Audio project [18]. The audio data in this project

consists of tapped telephone conversations made by the traders of the ENRON company and is publicly available on the internet [19].

This task differs from other reported experiments in several important ways. First, there are no prior target speaker models and there is no separate training data available for training speaker models. In fact, there is no demarcation between training and testing data and the entire database is utilized for both purposes. Given this database of telephone conversations, the task is to find conversations that have one or more speakers in common. This will establish links between the conversations and aid in the browsing of such a database. A user can select a particular conversation that he or she is interested in and the system should be able to automatically retrieve all the conversations that have speakers in common to the selected conversation.

Second, the poor quality of the audio data prevents us from using a typical approach to this problem. Typically speaker segmentation is used to separate the conversations into single-speaker portions. Speaker verification can then be employed to determine if the conversations have one or more speakers in common. Speaker segmentation, though useful, can sometimes be highly unreliable and may perform poorly due to factors such as rapid speaker interchange, background noise and poor audio quality. Speaker segmentation is usually done by detecting speaker changes in the conversation to form segments [20]. These segments are then combined using clustering algorithms to form single-speaker clusters. The number of clusters is usually chosen to be greater than the estimated number of speakers in the conversation. Though, this improves the purity of most of the clusters and may

yield an impure cluster containing speech data from more than one speaker; however the amount of speech data gets reduced in the pure clusters as some of it is lost in the residual clusters. This leads to lesser amount of speech data available for training and testing and can be the cause for poor performance. A more robust approach has been developed, where speaker segmentation was not used and the entire conversation was used to construct a multi-speaker model. Each conversation is used to train a multi-speaker model and speaker detection is performed on the remaining conversations in the database. The Switchboard-I database was used as a control database to test the performance of the algorithm. Besides, the completely automatic techniques, semi-automatic approaches have also been explored. These involve manual work and feedback from the user.

1.6 Outline of Thesis

The thesis is divided into five chapters. Chapter 2 describes the two databases that have been used in this study. The different voice mining techniques have been explained in detail in Chapter 3. Chapter 4 deals with the various experiments performed and the results obtained. Important conclusions and future work have been outlined in the Chapter 5 of the thesis.

Chapter 2

Databases

2.1 Overview

Two databases were used in this study: the Switchboard-I database and the ENRON database. The Switchboard-I corpus was collected in a sort of controlled fashion and a computer-driven "robot operator" system handled the calls, introducing a topic for discussion and recording the speech from the two subjects until the conversation was finished [21]. This database has high audio quality and was used as a control database to test the algorithms. The algorithms were then applied to the ENRON database. The collection of this database was not controlled and the audio quality is poor. The conversations were tapped and most of the speakers were not even aware that they were on a recorded line. A description of each database is given below.

2.2 Switchboard-I Database

The Switchboard-I (Release 2) corpus was collected by Texas Instruments, published by NIST and distributed by the Linguistic Data Consortium (LDC) [22]. This corpus is a collection of two-sided telephone conversations sampled at 8 kHz and it is designed so that no two speakers would converse together more than once.

The database consists of two channel conversations, where each channel contains one side of the conversation. The entire conversation can be obtained by summing the two channels. A subset consisting of 480 conversations was selected from the corpus. The choice of files ensured that there are at least 60 speakers (30 male and 30 female that were randomly chosen) who are present in 10 conversations. The remaining speakers are present in less than 10 conversations. The telephone conversations (considering only the speech part) are 1 to 9 minutes in duration. The ground truth for the voice mining task was constructed by identifying conversation pairs that have one speaker in common. This was not difficult to prepare since the speaker identification numbers of the participating speakers are provided in the Switchboard-I database.

A sample speech file from the Switchboard-I database is shown in Figure 2.1. The spectrograms of the Switchboard files are clean since the audio quality is high and there is very little background noise.

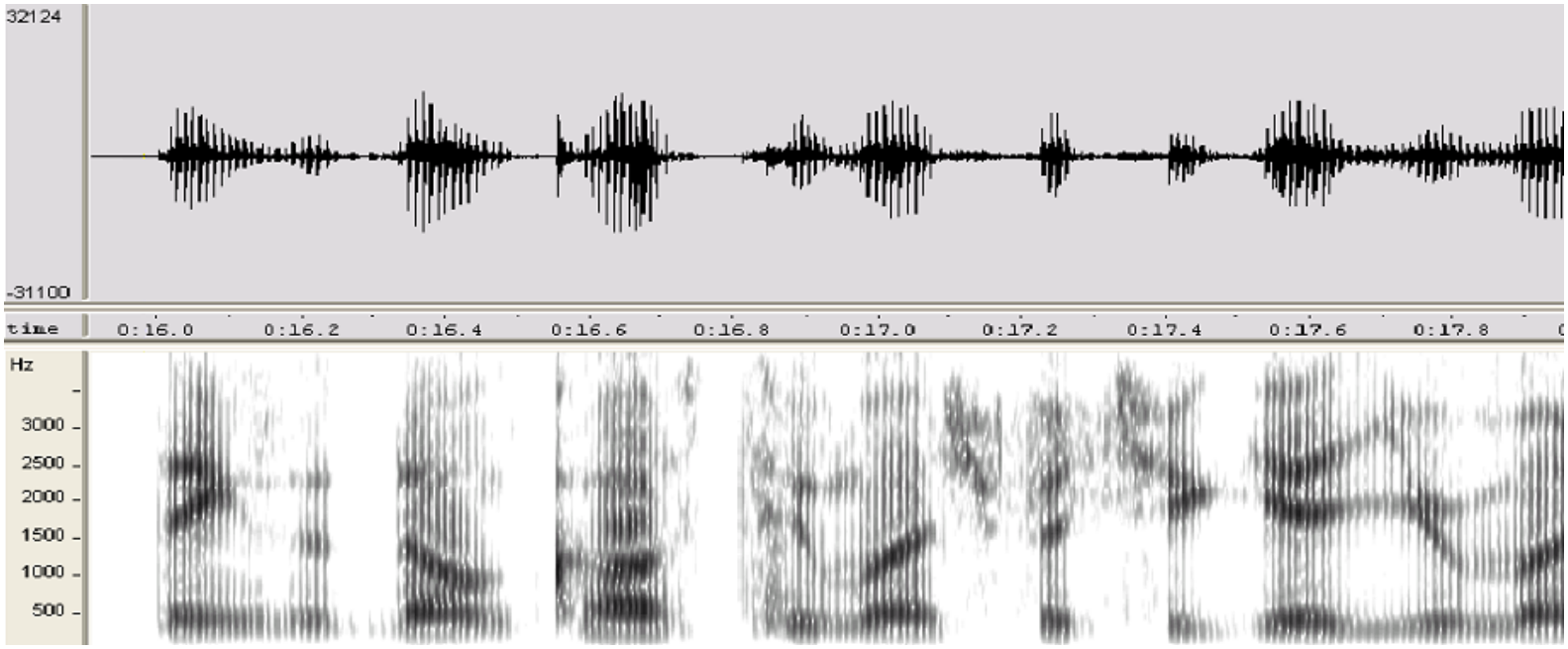

Figure 2.1: Portion of a sample Switchboard-I conversation: waveform (top panel) and spectrogram (bottom panel)

2.3 ENRON Database

The algorithms were evaluated on a portion of the ENRON Database. The ENRON speech database is publicly available on the internet [19] and consists of tapped telephone conversations made by the traders of the firm. This database is a part of the JIKD Email/Audio Project Database [18].

Some of the issues associated with this database are: (1) the audio quality is not high due to the presence of background noise, clipping of sound files and the presence of dial tones, telephone ringing and dialing sounds (2) there is rapid speaker interchange (3) speaker's voice varies with emotion (4) speakers talk in whispers (5) unbalanced conversations (6) length of the conversation varies (7) conversations include laughter, breathing noise, coughing, etc. (8) one wave file may contain multiple conversations (9) conference calls contain more than two speakers.

Most of the wave files that contain multiple conversations, have telephone sounds(dial tone, dialing or ringing) present between two conversations. These telephone sounds were automatically detected and were used to segment the wave file into individual conversations. However, a few wave files that do not have the telephone sound between two conversations were manually segmented. The telephone sounds were automatically removed from all the conversations so that they only contain speech. Since it is very difficult to build models using small conversations and to perform speaker detection, the algorithms were evaluated over conversations that are at least two minutes in duration. There are 156 conversations in the database that are at least two minutes in duration and they contain more than 125 unique

speakers. These files contain about one minute of speech per speaker, if there are two speakers in the file. The transcripts of these telephone conversations which contain the name of the speakers is also available and was used to prepare the ground truth for the voice mining task. Some of the transcripts do not contain the full name of the speaker and as a result there were ambiguities in the identity of some speakers. These ambiguities were resolved by listening to the conversations and manually comparing them.

A sample wave file from the ENRON database is shown in Figure 2.2. In this case the spectrogram is not as clean as the one for Switchboard. This is due to the poor audio quality of this database. The clipping of the data is also shown in the figure. Figure 2.3 shows a sample of the telephone sounds present in the ENRON database. The dial tone and dialing sounds of the telephone are shown in the figure. Such telephone sounds were automatically detected and removed.

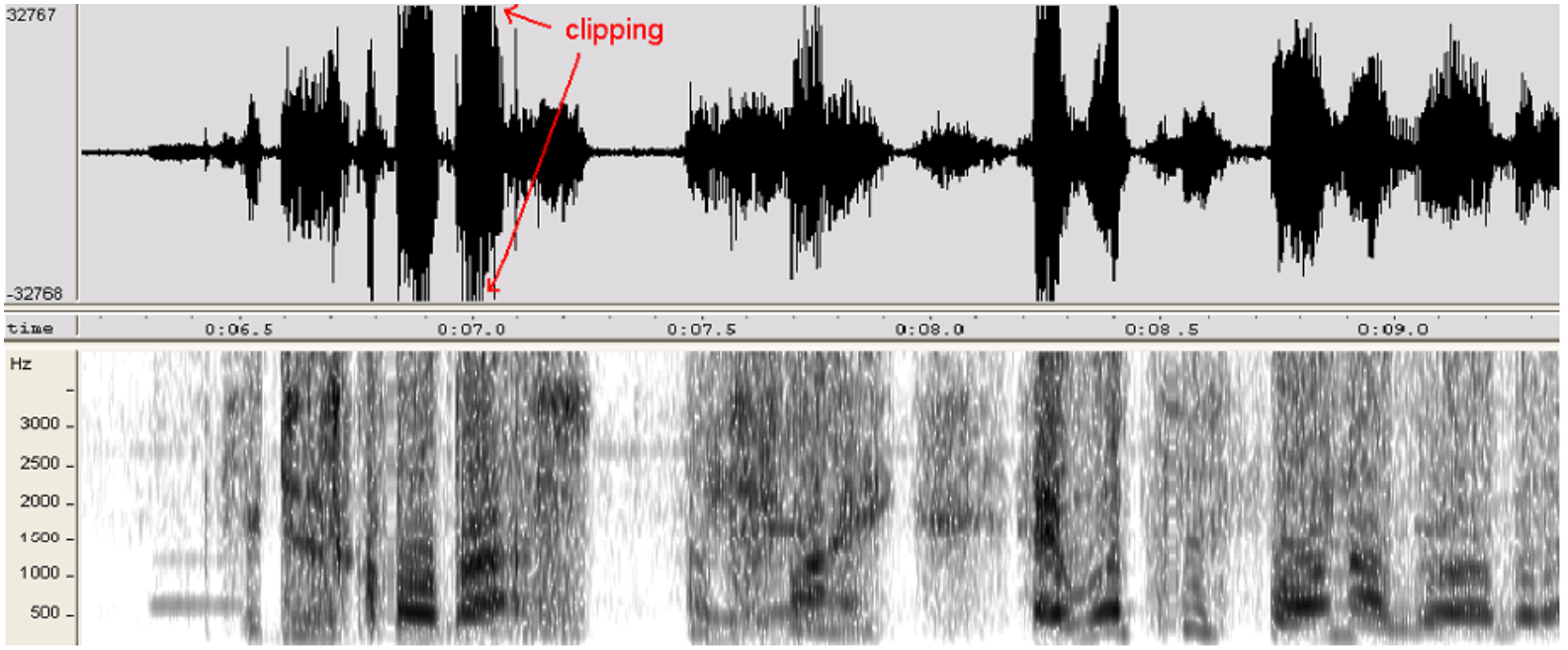

Figure 2.2: Portion of a sample ENRON conversation: waveform (top panel) and spectrogram (bottom panel)

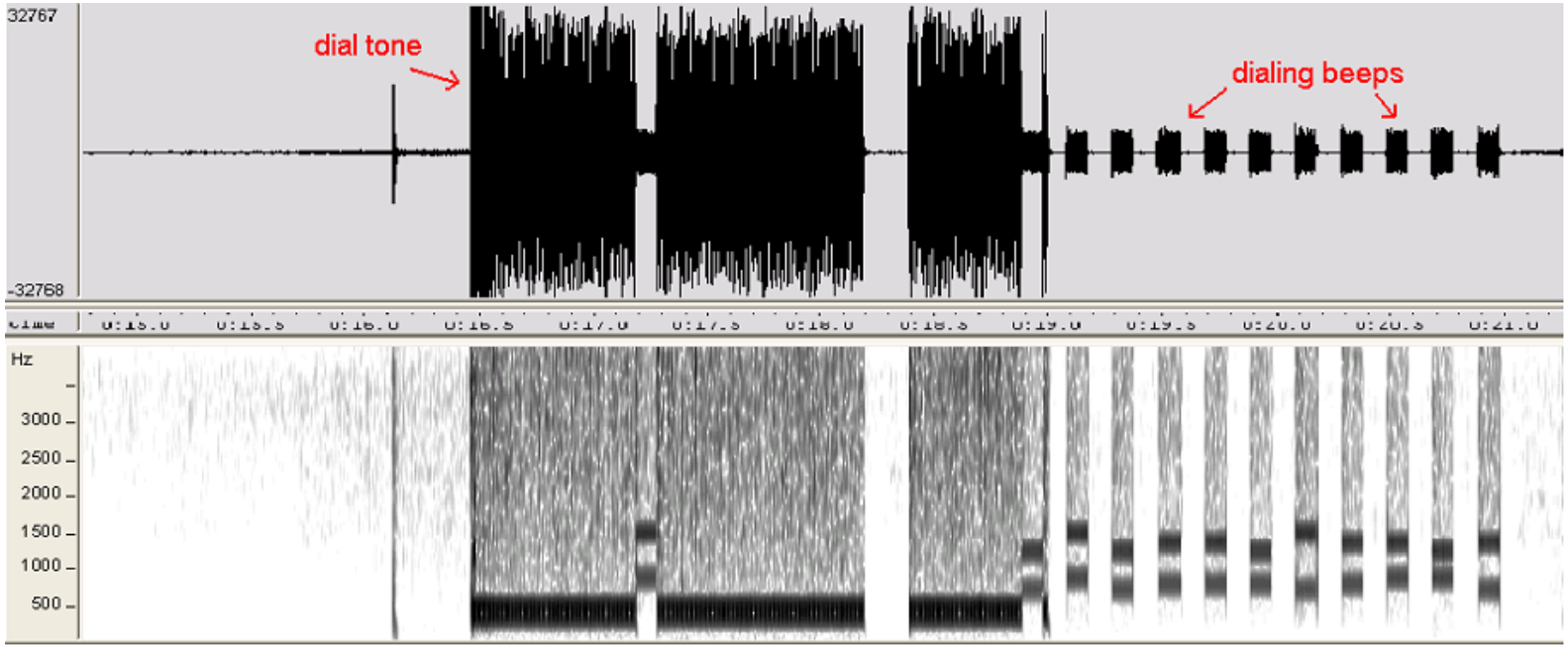

Figure 2.3: Telephone sounds in a sample ENRON conversation: waveform (top panel) and spectrogram (bottom panel)

Chapter 3

Voice Mining Techniques

3.1 Overview

Speaker detection is the task of determining whether a speaker is present in a given speech file. The speech file usually consists of two or more speakers. Speaker detection techniques may or may not involve speaker segmentation of the speech file [14]. In previous work done on speaker detection, it is usually assumed that the target speaker model or single speaker data for training the model is available. However, this is not the case in a voice mining scenario where given a database of telephone conversations, we wish to automatically extract as much information about the speakers as possible. Such information may include the number of speakers present in the database and identification of conversations that have one or more speakers in common. Subsequent manual intervention will be required to correct the errors made by the system or to obtain more specific information such as who is the common speaker in a conversation pair that have a speaker in common. Determining all the individuals that a person has conversations with can lead to the building of a social network of the speakers.

The above voice mining problem is complicated due to the fact that there are no target speaker models available and the training and testing data consists of multi-speaker speech files. However, in a real life scenario, this type of problem

is not an uncommon one. The task is to identify the conversation pairs that have one or more speakers in common and possibly estimate the number of speakers. However, we will not be able to automatically determine who the common speaker is.

There are different ways to approach this voice mining problem. One approach is to use automatic speaker segmentation to segment the multi-speaker speech files into single-speaker portions. Speaker verification can be applied to compare the single-speaker segments of two conversations to determine if the conversation pair has a speaker in common. Speaker segmentation, though useful, can sometimes be highly unreliable and may perform poorly due to factors such as rapid speaker interchange, background noise and poor audio quality. Manual effort may be required to optimize the speaker segmentation algorithm for a given database. In this study, we have attempted to use another approach that does not require speaker segmentation. A multi-speaker model is constructed using the entire unsegmented conversation and is used to find other conversations that have one or more speakers in common. There is no speaker segmentation performed on the training and testing file. These two approaches have been compared and a combination of the two approaches can be used to achieve better performance. These automatic approaches are not completely free of errors and a human needs to verify the decision made by the system. However, the automatic approaches will expedite the voice mining process by reducing the number of files that need to be verified manually.

In this voice mining scenario, we assume that each conversation takes place between two or more speakers. The task can be divided into two parts:

1. determine the conversations that have one speaker in common

2. determine the conversations that have two or more speakers in common

Note that we do not differentiate between the conversations that have two speakers in common and conversations that have three(or more) speakers in common. This is because it is difficult to determine the number of speakers that are in common and most of the conversations in the database contain only two speakers. The algorithms that have been developed approach the task in a slightly different way. The algorithms identify conversations that have at least one speaker in common and at least two speakers in common. The outputs of the algorithms can be combined to determine the conversations that have exactly one speaker in common and those that have two or more speakers in common.

3.2 The Algorithms for Voice Mining

This section deals with the different algorithms for voice mining of telephone conversations and is divided into two parts. The first part contains the algorithms that do not use speaker segmentation and the second part describes the algorithm that employs speaker segmentation for voice mining.

3.2.1 Algorithms without Speaker Segmentation

Two different algorithms were developed that do not use speaker segmentation. The first algorithm determines the conversation pairs that have at lease one speaker in common and the second algorithm identifies the conversations that have at least

two speakers in common. The outputs from the two algorithms can be used to determine the conversation pairs that have exactly one speaker in common, two or more speakers in common or no speaker in common.

In the absence of prior speaker models, every conversation file in the database is used once as training data and the remaining conversation files are tested against this conversation to determine if the files have a speaker in common.

3.2.1.1 At Least One Speaker in Common

Each conversation file is not segmented into its individual speakers, instead it is used to train a multi-speaker model of all the speakers present in the file. Since this model can be used to "verify" if the other speech files have at least one speaker in common with the training file, this task can be viewed in the speaker verification framework. As mentioned in Chapter 1, the Universal Background Model (UBM) of speech constructed using Gaussian mixtures has been extremely successful in speaker verification tasks [10]. In this study, each test file is compared against a multi-speaker model (based on a single conversation) and the UBM. The multi-speaker model is constructed by Maximum A Posteriori (MAP) adaptation of the UBM [10]. This multi-speaker model seeks to capture the characteristics of all the speakers present in the conversation. Usually the UBM is constructed from a large amount of speech data that is disjoint from the training and testing data. In the present study no such assumptions have been made about the availability of such data. Therefore, all files in the database were used to construct the UBM assuming

that the speech database is sufficiently large to give us a reasonably good background model of speech.

For normal speaker verification tasks, the entire test file is scored against the speaker model and the UBM as it contains the data of only one speaker. A likelihood ratio test is used to make the verification decision. However, in our case, the test file consists of a conversation of multiple speakers and we wish to solve the problem without speaker segmentation. Thus, a different technique is required to score the test files against the models. This can be accomplished by evaluating scores for segments of the speech file (which will probably contain speech corresponding to only one speaker as long as an appropriate window size can be chosen) rather than the entire file [11]. The score for the segment can be computed by averaging the frame based scores over the required segment. The duration of this averaging window and the amount of window overlap can be chosen empirically. Speaker verification accuracies are higher for test segments that are longer in duration since the instantaneous GMM scores are noisy and it is preferable to average over a longer time window. However if a long window is used, then it is very likely that the segment no longer contains the speech of only one speaker. Hence, there is a trade-off between the reliability of the average score of the segment and the probability that the segment contains speech from only one speaker. It is preferable in this case to have a long window which mainly contains a single speaker data because if even a single positive evidence of one speaker can be found, then it is sufficient to make the decision that the files have at least one speaker in common. The difference between the model score and the UBM score of each windowed segment are computed and

the maximum difference yields the likelihood ratio score(S) for the file:

$$S = \max_k \{log\ P(X_k|\lambda_1) - log\ P(X_k|\lambda_2)\} \tag{3.1}$$

where X_k is the k^{th} segment of the test file, λ_1 is the multi-speaker model, and λ_2 is the UBM. Since the verification is done on a segment of fixed duration, there is no need to do any score normalization to compensate for the length of the test utterance [23].

Each speech file from the given database is used to train a multi-speaker model and the above scoring technique is employed to score all the other files in the database against the model and the UBM. Let S_{ij} represent the likelihood ratio score when the i^{th} file is used for training the model and the j^{th} file for testing. For each pair of files, one file is used as a training file to score the other file and vice versa. We can take advantage of the fact that the two scores (S_{ij} and S_{ji}) are not identical. Different strategies can be used by taking the minimum, the maximum or the average of the scores to obtained a combined score $\tilde{S}_{ij}$. We can also combine the scores by always using the longer of the two files for training or for testing. This combined score is compared with a threshold to make the decision for the file pair (i, j):

$$\tilde{S}_{ij} = \max\{S_{ij}, S_{ji}\} \tag{3.2}$$

$$\tilde{S}_{ij} = \min\{S_{ij}, S_{ji}\} \tag{3.3}$$

$$\tilde{S}_{ij} = (S_{ij} + S_{ji})/2 \tag{3.4}$$

$$\tilde{S}_{ij} = S_{ij} \quad ...\ if\ length\ of\ i\ >\ length\ of\ j \tag{3.5}$$

$$\tilde{S}_{ij} = S_{ji} \quad ... \; if \; length \; of \; i \; > \; length \; of \; j \tag{3.6}$$

If $\tilde{S}_{ij} > \gamma$ then $H(i,j) = 1$ ... i^{th} and j^{th} file have at least 1 speaker in common

else $H(i,j) = 0$ ... i^{th} and j^{th} file have no speaker in common

where, γ = threshold and H is the hypothesis matrix representing the decisions made by the system. The value of the threshold can be set by using a few sample files, where the ground truth for these files is known or it is manually determined. The value of the threshold is usually set to achieve a fixed miss error rate or false alarm rate, depending on the application.

3.2.1.2 At Least Two Speakers in Common

The task of determining whether a conversation pair has at least two speakers in common is simpler than the previous task. Most of the telephone conversations in the database contain only two speakers and it is both the speakers that are common. However, it is assumed that the presence of additional speakers (usually the secretaries or people transferring the call) does not affect the algorithm adversely since these additional speakers are not the primary speakers and are present only for a small duration in the entire conversation. Once again, the problem fits into the speaker verification framework. A UBM is constructed from the available speech data. Each conversation file is used for training a multi-speaker model by MAP adaptation of the UBM. There is no need for speaker segmentation of the test files since most of the speech file contains the speakers of interest and the entire file can be scored. Similar to speaker verification the difference between the model score and

the UBM score of the test file are compared using a threshold to make the decision:

$$S = \{log\ P(X|\lambda_1) - log\ P(X|\lambda_2)\}/k \tag{3.7}$$

where X is the entire test file, λ_1 = multi-speaker model, λ_2 = UBM and k is number of frames in X. Note, that the score is obtained by dividing the difference in the likelihood scores by the number of frames in the test utterance. This compensates for the length of the test utterance [23], which varies depending on the test file.

Similar to the previous algorithm, we have two scores for each conversation pair since one file is used for training and the other for testing and vice versa. The combined score $\tilde{S}_{ij}$ can be used to make the decision if the conversation pair have two or more speakers in common:

If $\tilde{S}_{ij} > \gamma$ then $H(i,j) = 2$ ... i^{th} and j^{th} file have at least 2 speaker common

else $H(i,j) = 0$ ... i^{th} and j^{th} file have no speaker in common

where, γ = threshold and H is the hypothesis matrix representing the decisions made by the system.

The above algorithm is simpler since the entire conversation can be scored against the models instead of considering segments of the speech file.

3.2.2 Algorithm with Speaker Segmentation

In this approach, a separate speaker segmentation algorithm was first applied to split the telephone conversations into single-speaker portions. The speaker segmentation algorithm [20] uses the Bayesian Information Criterion (BIC) to detect speaker changes in the conversation and divides the conversation into single-speaker

segments. A clustering algorithm such as hierarchical clustering tries to combine the single-speaker segments in order to produce clusters that contain only single-speaker data. The threshold for the speaker change detection is set so that the number of misses are minimized even though there may be many false detections. The false detections can be corrected by the clustering algorithm which combines the similar segments to produce clusters that mainly contain single-speaker data.

However, some issues are encountered while using a speaker segmentation algorithm. Usually, the number of speakers in the conversation is not known a priori and it is difficult to estimate, so we do not know how many clusters the clustering algorithm should produce. The number of clusters is usually chosen to be greater than the estimated number of speakers. This improves the purity of some of the clusters at the cost of having some impure clusters. These impure clusters usually contain speech from both the speakers or they may contain simultaneous speech. However, it is also possible that a pure cluster may unnecessarily be broken down into two or more clusters. Since we are only interested in speaker detection, all the clusters can be scored against the model and we can take the maximum of the scores as the detection score. The speaker change detection algorithm using BIC works well on good quality data (such as the Switchboard-I database), but does not do a good job when applied to the ENRON database. It was found that using silence regions as points of speaker change is more robust and does a better job on the ENRON database.

Since most of the telephone conversations in the database have two speakers, the number of clusters was chosen to be three (one more than the estimated number

of speakers). Hence each file was automatically segmented into three clusters. To determine whether a conversation pair has at least one speaker in common, all the three clusters of one conversation were compared with all the three clusters of the other conversation. Let us consider the i^{th} file and j^{th} file in the database. Let i_1, i_2 and i_3 represent the three clusters of the i^{th} file and j_1, j_2 and j_3 represent the three clusters of the j^{th} file. If the i^{th} file is treated as the training file, then three models are trained using i_1, i_2 and i_3. Since the j^{th} file is considered as the testing file, j_1, j_2 and j_3 are tested against all the models. Since each cluster mainly contains single-speaker data, it is a speaker verification problem and we can score the entire file. Since we are not certain of the content of the three clusters, we have to consider all of them in the scoring process. The score from the likelihood ratio test for a particular training cluster and testing cluster is given as:

$$S_{i_p j_q} = \{log\ P(X|\lambda_1) - log\ P(X|\lambda_2)\}/k \tag{3.8}$$

where X is the cluster j_q , λ_1 = model for cluster i_p, λ_2 = UBM, k is the number of frames in X, $p = 1, 2, 3$ and $q = 1, 2, 3$.

The score S_{ij} is computed by taking the maximum of all the scores:

$$S_{ij} = \max_{p,q} S_{i_p j_q} \qquad where\ p = 1, 2, 3 \ \ q = 1, 2, 3 \tag{3.9}$$

Similar to the previous algorithm, the score $\tilde{S}_{ij}$ is obtained by taking a combination of the scores S_{ij} and S_{ji}. This combined score is compared with a threshold to make the decision for the file pair (i, j):

If $\tilde{S}_{ij} > \gamma$ then $H(i, j) = 1$ ... i^{th} and j^{th} file have at least 1 speaker in common

$$\text{else } H(i,j) = 0 \text{ ... } i^{th} \text{ and } j^{th} \text{ file have no speaker in common}$$

where, γ = threshold and H is the hypothesis matrix representing the decisions made by the system.

However, we are not certain about the contents of each cluster and it is possible that two clusters contain single-speaker data of the same speaker. Hence, with this system it is difficult to determine the number of speakers that are in common. As a result, only the decision that the files have at least one speaker in common is taken. The algorithm cannot be used to determine if the conversations have at least two speakers in common.

3.3 Methodology

For speaker detection only the speech portion of the audio file should be used. The silence portion of the conversation was discarded using an energy-based threshold. The average energy of a frame in the utterance was computed and frames that are less than η times this average are discarded. A 25 ms frame size was used and the threshold η was manually adjusted using a few sample files. The telephone sounds in the ENRON database were automatically removed from the audio files using a detector for telephone sounds. The detector takes advantage of the fact that the telephone sounds appear as strong continuous tracks in the spectrum of the signal. These tracks exhibit very little deviation across time unlike formants that show more variability as the formant tracks slowly change with time. This difference was used to detect the presence of telephone rings and dial tones.

The speech files of the databases were parameterized using Mel Frequency Cepstral Coefficients (MFCCs). Nineteen MFCCs and their derivatives were computed every 10 ms to give 38 parameters per frame. Cepstral Mean Normalization (CMN), RASTA filtering [24] and warping of the coefficients to zero mean and unit variance [25] were used to achieve improved channel compensation. The first step is to construct the UBM using the available data. Since using all the speech files in the database would give an extremely large number of frames, the frames were downsampled. The downsampling factor was automatically adjusted for each file, so that each file contributes approximately the same amount of data to the UBM. Thus, the downsampling factor is larger for files that are longer in duration and they do not dominate the UBM. The average downsampling factor for the ENRON database was four. The UBM was trained using the MFCCs of the downsampled frames. The MIT-LL GMM system [10] was used for the experiments. Various model orders, which determine the number of Gaussian mixtures in the UBM were tried. The algorithms for determining if the conversation pair have at least one speaker in common or at least two speakers in common were tested. An averaging window was used to smooth the likelihood scores of the model and the UBM. Windows of duration 2 seconds to 14 seconds were tried and we usually used a 50% overlap of the window.

A ground truth matrix or reference matrix (R) was prepared that has dimensions N x N, where N is the number of speech files in the database. The elements of this matrix are tertiary-valued:

$R(i, j) = 0$ ... if the i^{th} and j^{th} files have no speaker in common

$R(i,j) = 1$ ... if the i^{th} and j^{th} files have exactly one speaker in common

$R(i,j) = 2$ ... if the i^{th} and j^{th} files have two or more speakers in common

In the above equations $i \neq j$, since we are not interested in the diagonal elements of R. The R matrix is symmetric and one matrix for each database was constructed. The hypothesis matrix (H) is usually symmetric and was computed using the different algorithms described in Section 3.2. The two algorithms (at least one speaker in common and at least two speakers in common) were tested separately. The hypothesis and reference matrix were compared to compute the miss error rate and false alarm error rate.

Assuming it is possible to obtain a perfect hypothesis matrix, it is possible to estimate the number of speakers in the database using graph theory algorithms. This estimate will be more accurate if all the conversations in the database consist of two speakers. However, if the conversations contain more than two speakers, we will be underestimating the number of speakers. The perfect hypothesis matrix will be equal to the reference matrix and each element will be tertiary valued as explained above.

Let us assume, that there are N conversations in the database and all contain two speakers. The element $H(i,j)$ (or $H(j,i)$) of the hypothesis matrix indicates if the i^{th} and j^{th} conversation in the database have one speaker, two speakers or no speaker in common. The conversations can be considered as the nodes of an undirected weighted graph, which contains N nodes. An edge indicates that the conversation pair has at least one speaker in common and the weight of the edge (one or two) specifies the number of speakers that are in common. When two nodes are

connected by an edge having a weight of two, the nodes contain the same speakers. Since these nodes are nearly identical in the graph(they have the same edges with the other nodes in the graph), one of the nodes can be collapsed. However, the situation is not so trivial when the nodes are connected by an edge having a weight of one.

A maximal complete subgraph (clique) is a complete subgraph that is not contained in any other complete subgraph. Thus, for every two nodes in a clique, there exists an edge connecting the two. Here, we are considering edges having a weight of one, since the nodes connected with edges of weight two have already been collapsed. Locating the cliques in the graph is important since these cliques contain all files that have at exactly one speaker in common if the size of the clique is greater than three. For cliques of size three, containing a weight of one on each edge, the number of speakers can be three or four. This ambiguity cannot be resolved and there can be an error of one for every clique of size of three (see Figure 3.1). For cliques of size k, where k is greater than three, the number of speakers in the clique is k+1 (see Figure 3.2). Algorithms have been developed in graph theory for finding all the cliques of an undirected graph [26]. The entire graph may contain a number of cliques of different sizes. To obtain an estimate of the number of speakers in the database, all the cliques have to be taken into account and the connections between the cliques also have to be considered. Each isolated node in the graph (that are not connected to any other node) will contribute two speakers to the estimate of the number of speakers.

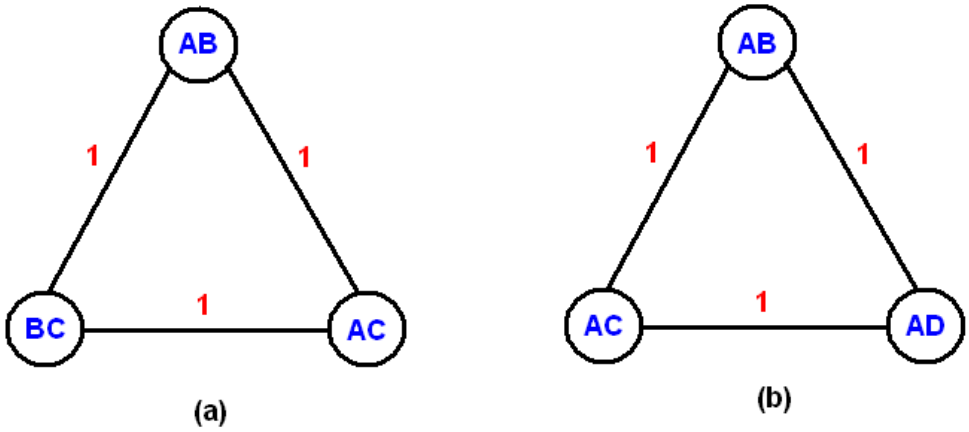

Figure 3.1: Ambiguity in cliques of size three: (a) No. of speakers = 3 (b) No of speakers = 4. The weights of each edge is marked in red.

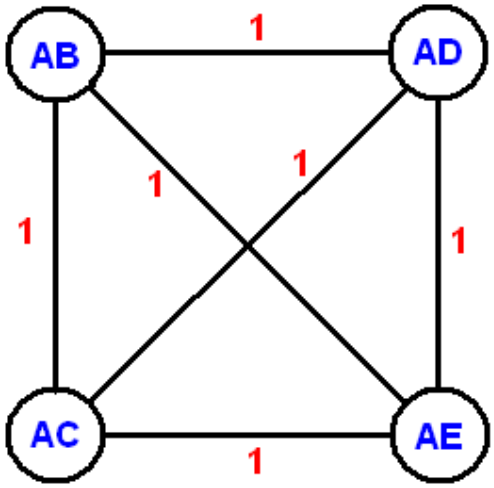

Figure 3.2: Cliques of size four with each edge having weight one (marked in red). No. of speakers = 5.

Chapter 4

Experiments and Results

4.1 Overview

The algorithms for identifying the conversation files that have at least one speaker in common and at least two speakers in common were tested. The Switchboard-I database was used as a control database to evaluate the voice mining techniques. Each conversation in the Switchboard-I database contains two speakers and the database was designed in such a way that two speakers do not converse more than once. As a result, there are no conversation pairs that have both speakers in common and we were unable to test the algorithm for at least two speakers in common on this database. However, the ENRON database does not have this problem and we were able to test both types of algorithms on this database. The results of the experiments are discussed in this chapter. The results on the Switchboard-I database are presented first, followed by the results on the ENRON database. The performance of the algorithms is characterized using the standard Detection Error Tradeoff (DET) curves [2]. Equal Error Rate (EER) is also used as a performance metric. The EER point is marked with a red circle on the DET curve.

4.2 Results on Switchboard-I Database

A portion of the Switchboard-I database was used. The statistics of this test database are given in Table 4.1. The positive instances in the table represent the conversation pairs having one speaker in common and the negative instances are the conversation pairs that have do not have a speaker in common. For the positive instances, $R(i,j) = 1$ and $R(j,i) = 1$ and for the negative instances $R(i,j) = 0$ and $R(j,i) = 0$, where R is the ground truth matrix.

Table 4.1: Statistics of the database for one speaker in common

No. of speech files	480
No. of speakers	186
Duration of speech files	1-9 min
No. of positive instances	3556
No. of negative instances	111404

4.2.1 Effect of UBM Model Order

The model order of the UBM has a significant impact on the performance of the algorithm. This is demonstrated by varying the model order while using the same score combination (average score) and a window duration of 8 seconds for all experiments (Figure 4.1). Increasing the model order leads to improved performance, however the computational complexity of the algorithm increases as well. The best EER achieved is 8.87% with a a UBM model order of 2048, an

average score combination and a 8 seconds window duration.

Table 4.2: Equal error rate for different model order of UBM

Model Order of UBM	**Equal Error Rate (EER)**
256	12.91
512	11.32
1024	9.73
2048	8.87

4.2.2 Effect of Window Duration

A sliding window was used to detect the presence of a speaker. A 50% window overlap was used and to observe the effect of the window duration on the performance, the window length was varied from 2 seconds to 30 seconds. The other factors such as the score combination technique and the model order of the UBM were kept constant. The average score combination was utilized and a 2048-mixture GMM was used. Figure 4.2 illustrates that the algorithm performs better for a window duration of 8 seconds, however it is not very sensitive to the window duration parameter. For convenience, only the best case (8 sec) and a few other cases (2,14 and 30 sec) are shown. The DET curves of the intermediate window durations lie between the 2 seconds and 30 seconds DET curves.

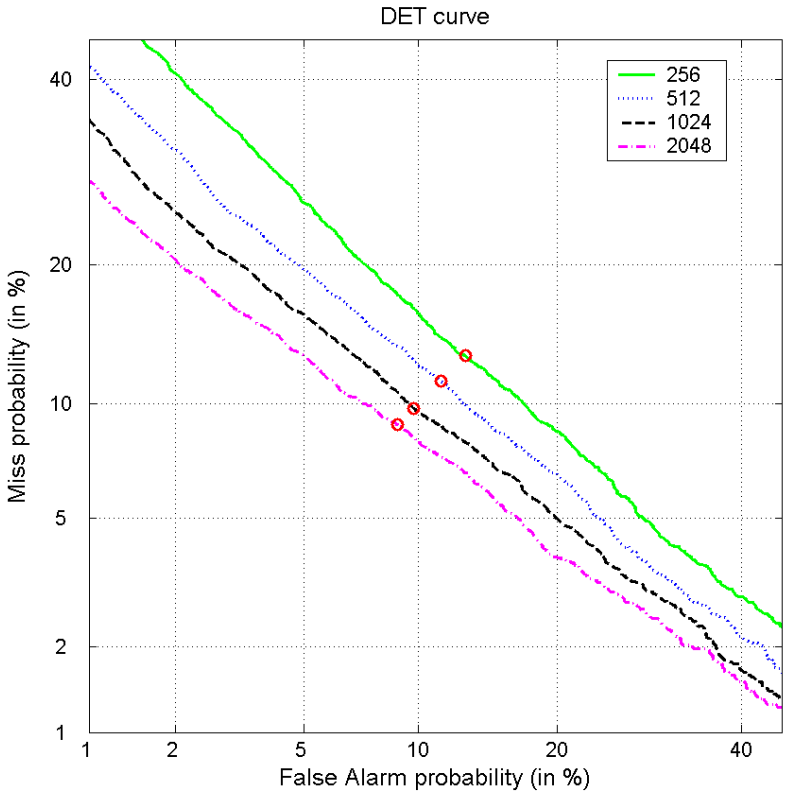

Figure 4.1: 1 speaker in common: Different model order of UBM (256,512,1024 and 2048). The EER point is marked with a red circle.

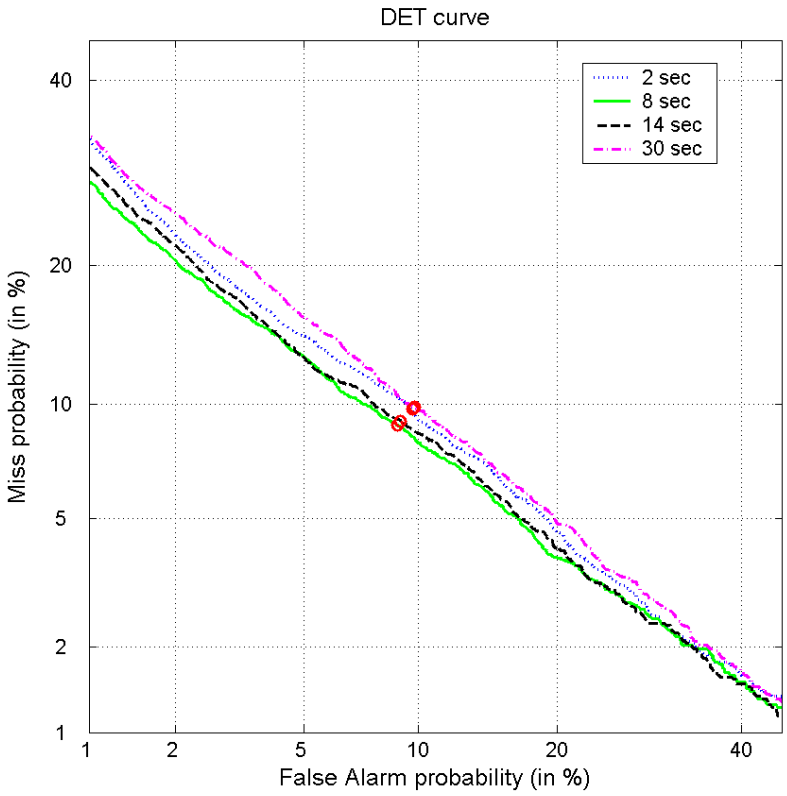

Figure 4.2: 1 speaker in common: Different window durations (2, 8, 14 and 30 sec). The EER point is marked with a red circle.

Table 4.3: Equal error rate for different window durations

Window Duration (sec)	**Equal Error Rate (EER)**
2	9.70
8	8.87
14	9.06
30	9.81

4.2.3 Different Score Combinations

Five score combination techniques were employed to combine the scores S_{ij} and S_{ji}: (a) mean (b) minimum (c) maximum (d) using the longer utterance for training and (e) using the longer utterance for testing. The score combinations were evaluated by holding the number of mixtures of the GMM and the window duration constant. A 2048-mixture GMM was used and the window duration was 8 seconds. Figure 4.3 shows the Detection Error Tradeoff (DET) curves for the different scoring strategies. The figure shows that taking a mean of the two scores gives the best results. The mean score combination is compared to no combination of the scores in Figure 4.4. The EER values are given in Table 4.4.

4.2.4 Using the N-best Scores

Instead of considering only the most positive evidence of a speaker, which is characterized by the maximum score of all the windows, the N-best scores can be considered. Thus, we are searching for more than just one evidence of the speaker

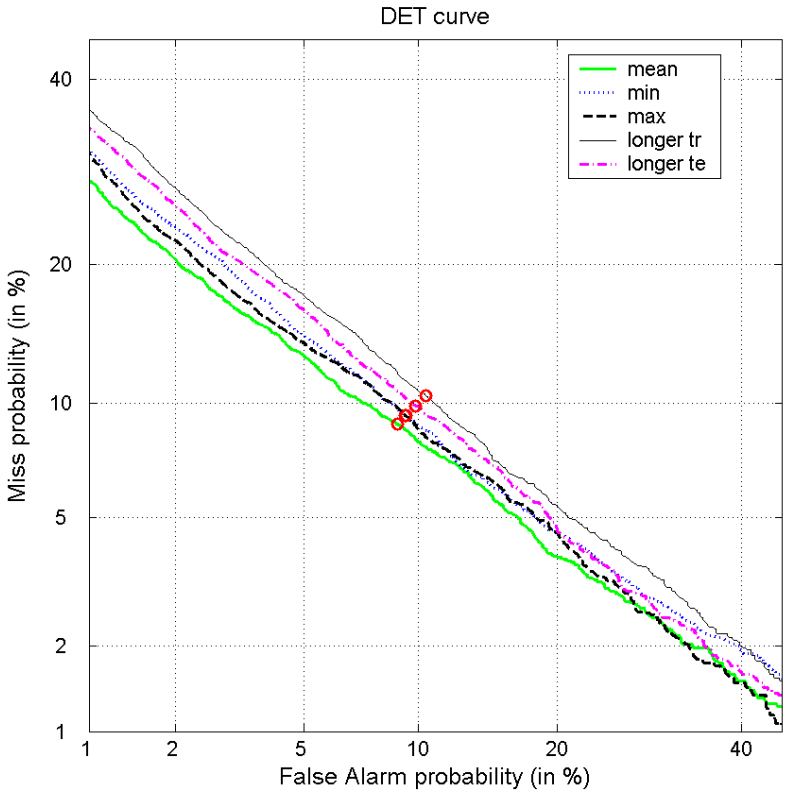

Figure 4.3: 1 speaker in common: Different score combinations (mean, minimum, maximum, longer training utterance and longer test utterance). The EER point is marked with a red circle.

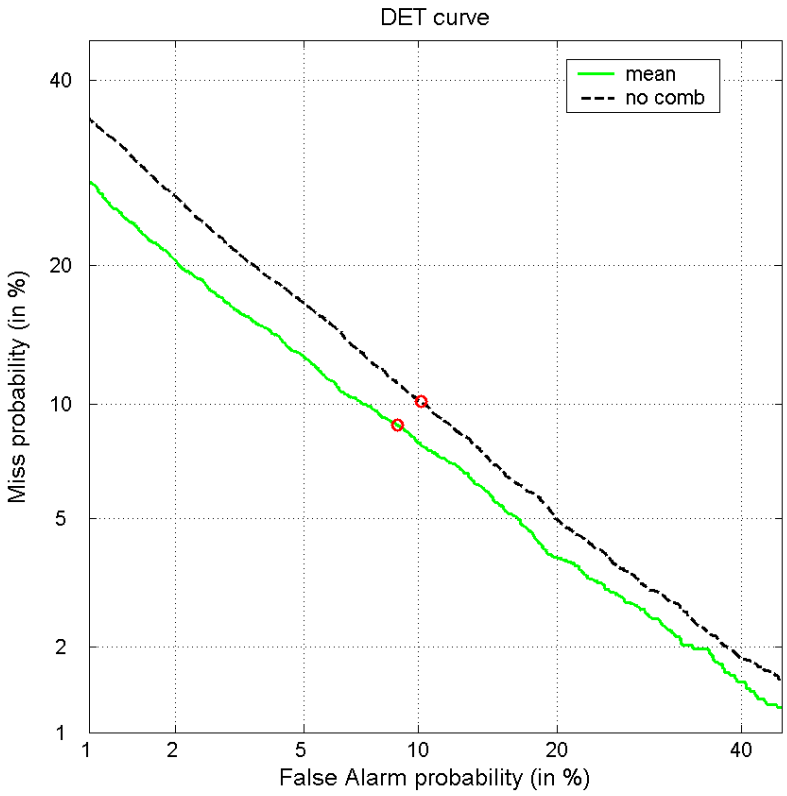

Figure 4.4: 1 speaker in common: Comparing the best score combination (mean) with no combination. The EER point is marked with a red circle.

Table 4.4: Equal error rate for different score combinations

Score Combination	**Equal Error Rate (EER)**
Mean	8.87
Minimum	9.30
Maximum	9.31
Longer train	10.41
Longer test	9.84
No combination	10.14

in the conversation. An average of the N-best scores was used and the results are given in Table 4.5. A 2048-mixture GMM was used and the window duration was 8 seconds. The EER values indicate that the average of the 3-best scores yields the lowest EER of 8.74%. However, the improvement is not very significant and for most of the experiments we have employed the 1-best score.

Table 4.5: Equal error rate for mean of N-best scores

Score Used	**Equal Error Rate (EER)**
1-best	8.87
2-best	8.86
3-best	8.74
4-best	8.80
5-best	8.89

4.2.5 Comparison with Perfect Speaker Segmentation

The performance of the algorithm was compared against the ideal case where perfect speaker segmentation can be achieved. The Switchboard-I database consists of two channel conversations, where each channel contains one side of the conversation. Perfect speaker segmentation was simulated by using each channel, which contains only single-speaker data. There is a significant improvement in the performance as shown by the DET curve (Figure 4.5). In both cases a 1024-mixture UBM was used. The EER of the system without speaker segmentation is 9.73% while the EER for a system that can achieve perfect speaker segmentation is 6.05%. However, the distance between the DET curves decreases for the region with miss probability less than 2%. Since perfect speaker segmentation is the ideal case, this result gives an upper ceiling of the system's performance. To obtain further improvements in performance, we need to improve the speaker detection engine or use better features.

4.3 Results on ENRON Database

The experiments were performed on a portion of the ENRON database. This test database was divided into three sets according to the duration of the conversations: conversations that are longer than (a) 1 minute, (b) 2 minutes and (c) 3 minutes form Set I, Set II and Set III, respectively. The results for the algorithms for at least one speaker in common are presented first, followed by the results for at least two speakers in common.

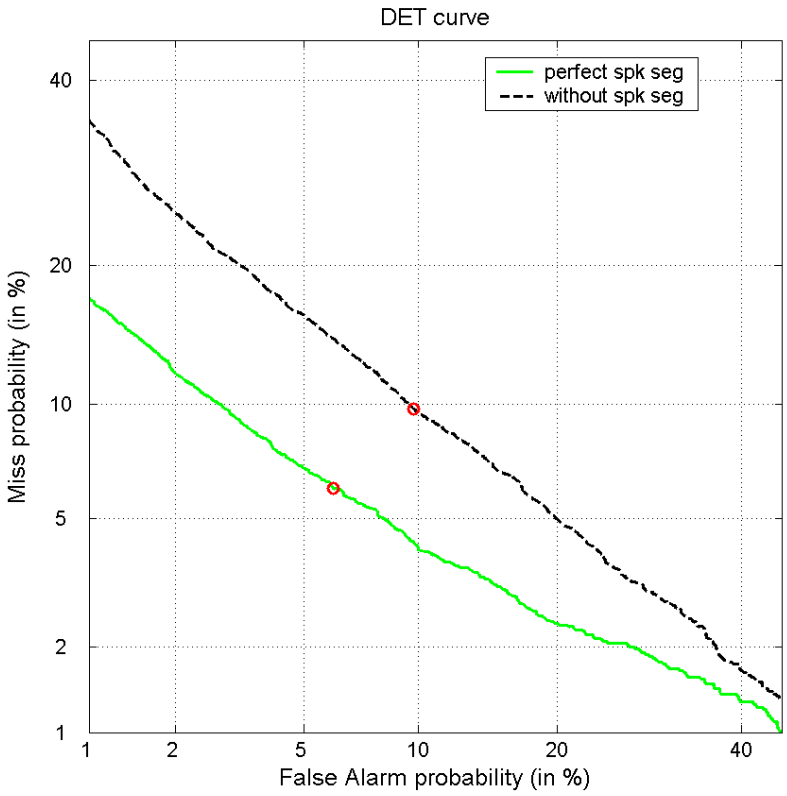

Figure 4.5: 1 speaker in common: With perfect speaker segmentation and without speaker segmentation. The EER point is marked with a red circle.

4.3.1 At Least One Speaker in Common

The algorithms using speaker segmentation and without speaker segmentation were both evaluated. The statistics of the 3 sets of this database are summarized in Table 4.6. For the positive instances, $R(i,j) = 1$ (exactly one speaker in common) or $R(j,i) = 2$ (two or more speakers in common) and for the negative instances $R(i,j) = 0$ (no speaker in common), where R is the ground truth matrix. Since the R matrix is symmetric ($R(i,j) = R(j,i)$), we need to consider only half of the matrix.

Table 4.6: Statistics of the database for at least one speaker in common

Statistics	Set I	Set II	Set III
Min. conversation length	1 min	2 min	3 min
No. of speech files	242	156	112
No. of speakers	157	125	102
No. of positive instances	2047	1048	635
No. of negative instances	27114	11042	5581

Similar experiments were performed on the ENRON database to determine the effect of the UBM model order, window duration and the different score combinations. The results were consistent with the trends observed on the Switchboard-I database. However, increasing the model order of the UBM does not improve the accuracies significantly and a UBM of 1024 mixtures was used for all the experiments. The window duration of 8 seconds yields the best results and this window with a

50% overlap was employed in all the experiments. The different score combinations are discussed below.

4.3.1.1 Without Speaker Segmentation

The results of the algorithm for at least one speaker in common without speaker segmentation are given in this section.

Different Score combinations

Three score combination techniques were employed to combine the scores S_{ij} and S_{ji}: (a) mean (b) minimum and (c) maximum of the two scores. The score combinations were evaluated by holding the number of mixtures of the GMM and the window duration constant. A 1024-mixture GMM was used and the window duration was 8 seconds. Figure 4.6 shows the Detection Error Tradeoff (DET) curves for the different scoring strategies. Similar to results obtained on the Switchboard-I, taking a mean of the two scores gives the best results. There is an improvement by about 10% in the EER compared to not using any score combination.

Table 4.7: Equal error rate for different score combinations

Score Combination	**Equal Error Rate (EER)**
Mean	18.70
Maximum	19.05
Minimum	19.94
No Combination	20.75

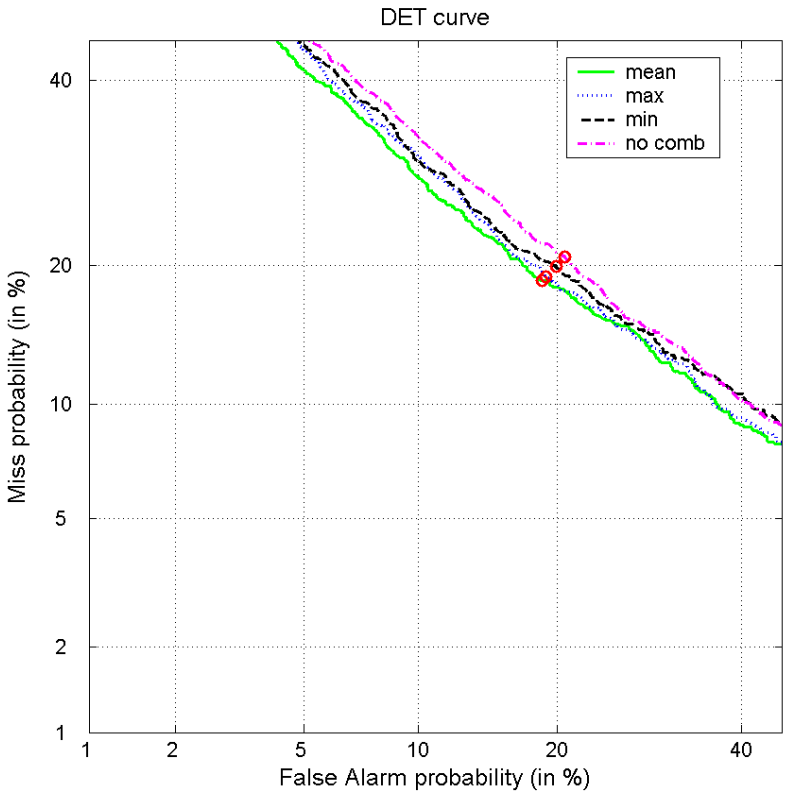

Figure 4.6: At least 1 speaker in common: Different score combinations. Conversation length > 2 min. The EER point is marked with a red circle.

Different Conversation Lengths

The algorithm was evaluated on all the three sets of the database. The minimum length of the conversation in the sets are: (a) 1 minute, (b) 2 minutes and (c) 3 minutes. Using longer conversations provides more training and testing data to the algorithm. As expected, the performance of the algorithm improves with increasing conversation lengths.

Table 4.8: Equal error rate for different conversation lengths

Min. Conversation Length	**Equal Error Rate (EER)**
1 min	21.62
2 min	18.70
3 min	16.38

4.3.1.2 With Speaker Segmentation

The results of the algorithm for at least one speaker in common with automatic speaker segmentation are given in this section.

Different Score combinations

Three score combination techniques were employed to combine the scores S_{ij} and S_{ji}: (a) mean (b) minimum and (c) maximum of the two scores. The score combinations were evaluated by holding the number of mixtures of the GMM and the window duration constant. A 1024-mixture GMM was used and the window duration was 8 seconds. Figure 4.6 shows the Detection Error Tradeoff (DET) curves

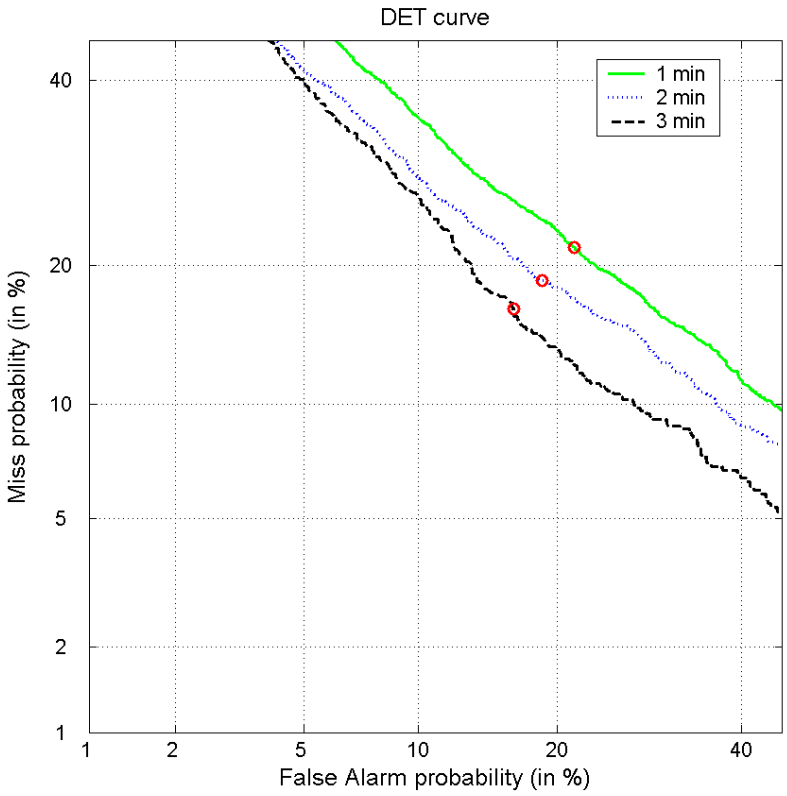

Figure 4.7: At least 1 speaker in common: Different conversation lengths. The EER point is marked with a red circle.

for the different scoring strategies. The DET curve and the EER values(Table 4.9) show that there is no appreciable difference between various score combinations and no combination. The mean score combination was used for all the following experiments.

Table 4.9: Equal error rate for different score combinations

Score combination	**Equal Error Rate (EER)**
Mean	21.66
Maximum	21.67
Minimum	21.18
No Combination	21.71

Different number of clusters used

The speaker segmentation algorithm produces three clusters for each conversation. The clusters were sorted according to the amount of speech data in each cluster. The smallest cluster is the problematic one. This cluster may be of comparable size to the other two clusters and may contain single-speaker data (pure cluster) or multi-speaker data (impure cluster). In this experiment, we eliminated the third cluster of each file and used the two larger clusters. However, the results indicate that the third cluster is important and it is better to use all three clusters for the algorithms involving speaker segmentation (see Table 4.10 and Figure 4.9).

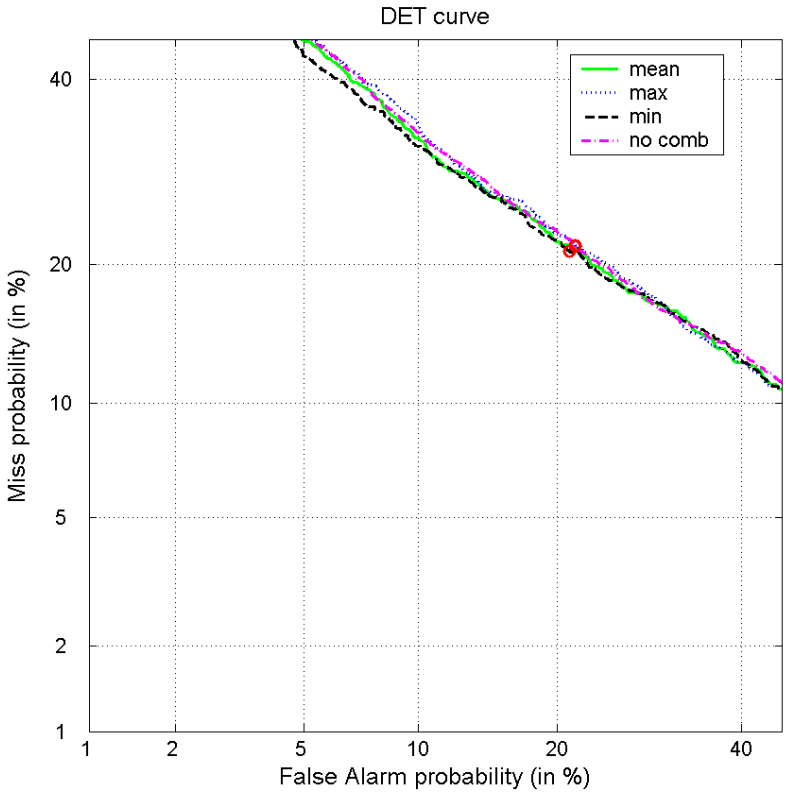

Figure 4.8: At least 1 speaker in common: Different score combinations with speaker segmentation. Conversation length > 2 min. The EER point is marked with a red circle.

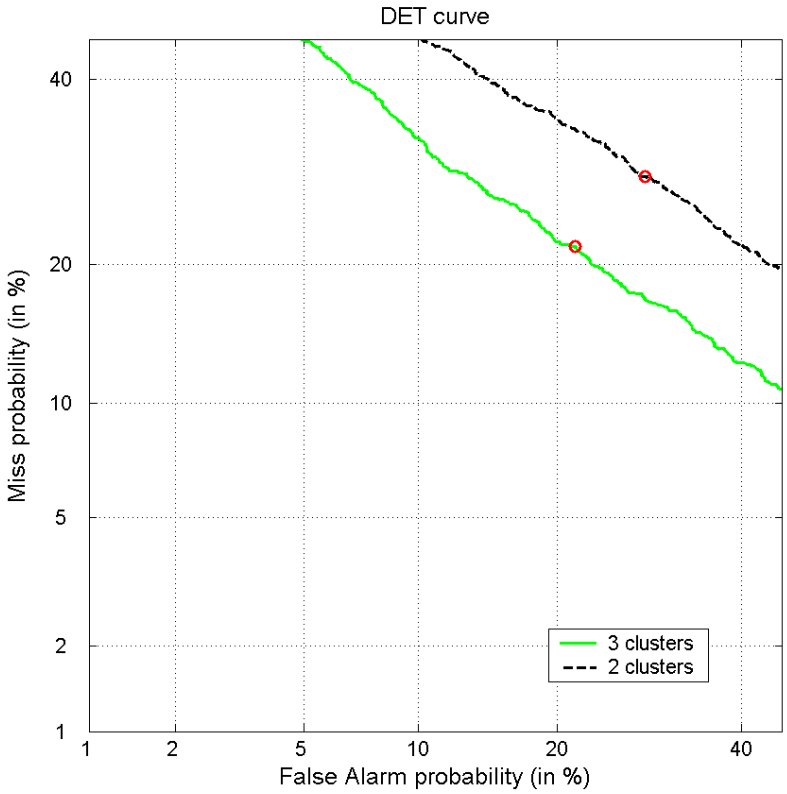

Figure 4.9: At least 1 speaker in common: Different number of clusters used with speaker segmentation. Conversation length > 2min. The EER point is marked with a red circle.

Table 4.10: Equal error rate for different number of clusters used

No. of Clusters	**Equal Error Rate (EER)**
2	28.72
3	21.66

4.3.1.3 Comparison of With and Without Speaker Segmentation

The two different algorithms (with and without speaker segmentation) for at least one speaker in common are compared. The DET curves and the EER values clearly indicate that the algorithm without speaker segmentation outperforms the algorithm that uses speaker segmentation. This is mainly due to the fact that using the speaker segmentation algorithm decreases amount of training data for building a speaker model. The speaker segmentation algorithm splits the conversation into three clusters and a model is trained on each cluster as opposed to using the entire conversation to train a multi-speaker model. A simple strategy was employed to combine the output of the two algorithms: the score of each algorithm was normalized to a value between zero and one and the mean of the normalized scores was used. There is a marginal improvement by combining the two algorithms in the EER region. The improvements are more pronounced for lower miss probability rates as seen in Figures 4.10 and 4.11.

If we would like to reduce the errors made by the system, a semi-automatic approach can be adopted. The output of the system can be manually verified to discard false detections made by the system. The advantage of this approach is that

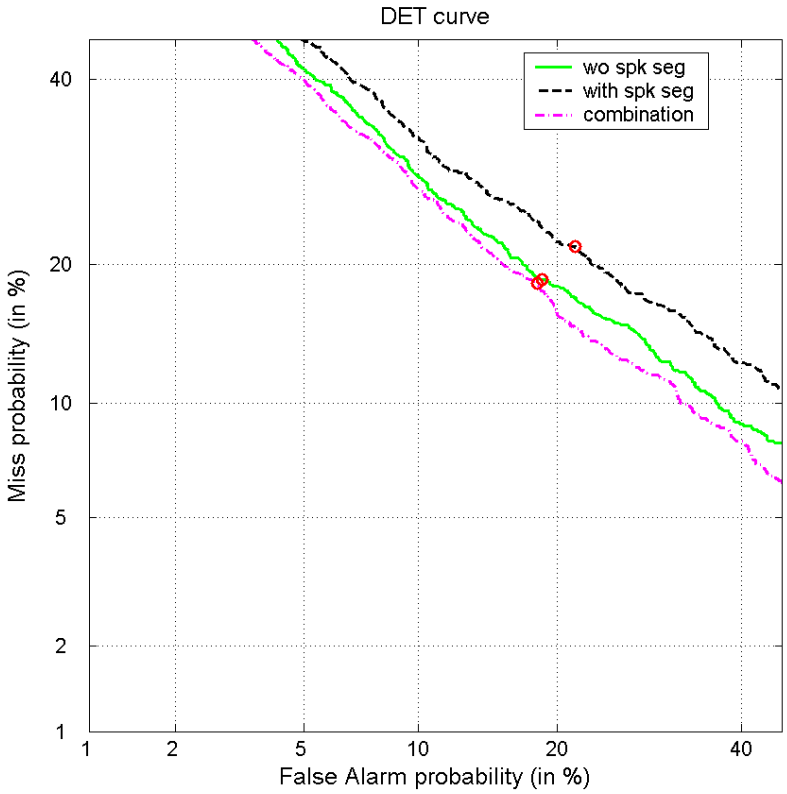

Figure 4.10: At least 1 speaker in common: With and without speaker segmentation and combination of the two. Conversation length > 2 min. The EER point is marked with a red circle.

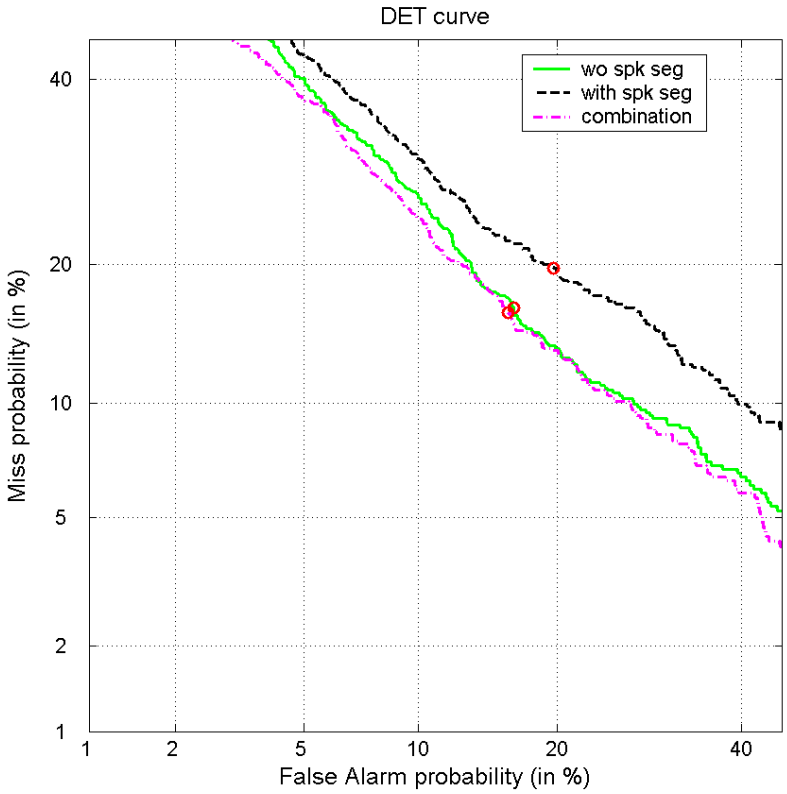

Figure 4.11: At least 1 speaker in common: With and without speaker segmentation and combination of the two. Conversation length > 3 min. The EER point is marked with a red circle.

Table 4.11: Equal error rate for different algorithms

Algorithm	**Length > 2 min**	**Length > 3 min**
Without spk seg	18.70	16.38
With spk seg	21.66	19.69
Combination	18.32	15.98

manual effort is considerably reduced and the user has to check a portion of the entire database. In this approach, only the false alarm errors can be corrected and we cannot correct the miss errors. Thus, it is desirable to operate the system at a low miss probability rather than using the system at the EER point. For example, if we use a combination of the two algorithms (with and without speaker segmentation) on set-III of the ENRON database, for a miss probability of 5%, the false alarm probability is 42%. Thus, the user need not check 58% of the data and needs to verify the remaining portion. If we can afford to relax the miss probability to 10%, the false alarm probability is 27%. In this case, 73% of the database need not be manually checked. Thus, there is a tradeoff between the manual effort required and the number of miss errors that will occur.

4.3.2 At Least Two Speakers in Common

The algorithm for detecting at least two speakers in common without using speaker segmentation was evaluated. The statistics of the 3 sets of this database are summarized in Table 4.12.

Table 4.12: Statistics of the database for at least two speakers in common

Statistics	Set I	Set II	Set III
Min. conversation length	1 min	2 min	3 min
No. of speech files	242	156	112
No. of speakers	157	125	102
No. of positive instances	141	74	56
No. of negative instances	29020	12016	6160

4.3.2.1 Different Score combinations

Three score combination techniques were employed to combine the scores S_{ij} and S_{ji}: (a) mean (b) minimum and (c) maximum of the two scores. The score combinations were evaluated by holding the number of mixtures of the GMM and the window duration constant. A 1024-mixture GMM was used and the window duration was 8 seconds. The DET curve (Figure 4.12) shows that taking a mean of the two scores gives the best results. The DET curves for at least two speakers in common are not as smooth as the curves obtained for at least one speaker in common. The step-like nature of the curve is due to the small number of positive instances.

4.3.2.2 Different Conversation Lengths

The algorithm was evaluated on all the three sets of the database. The minimum length of the conversation in the sets are: (a) 1 minute, (b) 2 minutes and (c)

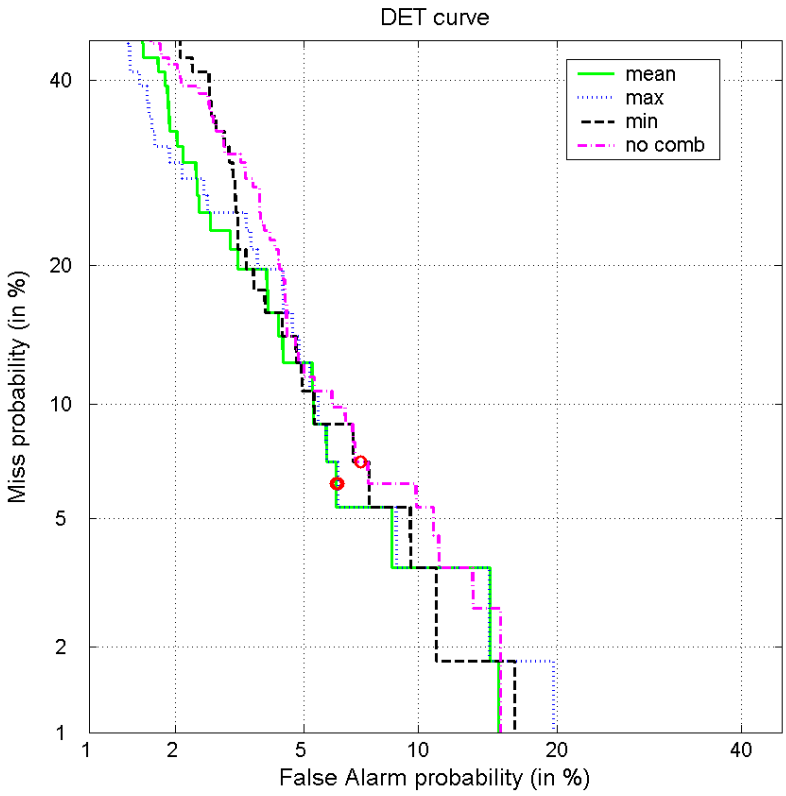

Figure 4.12: At least 2 speakers in common: Different score combination. Conversation length > 3 min. The EER point is marked with a red circle.

Table 4.13: Equal error rate for different score combinations

Score Combination	**Equal Error Rate (EER)**
Mean	6.25
Maximum	6.25
Minimum	7.14
No Combination	7.14

3 minutes. There is a marginal increase in the EER for the second set as compared to the first set. This probably due to the small number of positive instances in the database. Conversations that are at least 3 minutes in duration yield the lowest EER indicating that increasing the amount of training and testing data helps the algorithm.

Table 4.14: Equal error rate for different conversation lengths

Min. conversation length	**Equal Error Rate (EER)**
1 min	8.51
2 min	8.97
3 min	6.25

Most of the errors made by the algorithm for at least two speakers in common are due to the conversation pairs having exactly one speaker in common. The presence of this speaker can increase the score of the entire conversation and leads to false alarm errors. The EER for conversations greater than 3 minutes in duration is

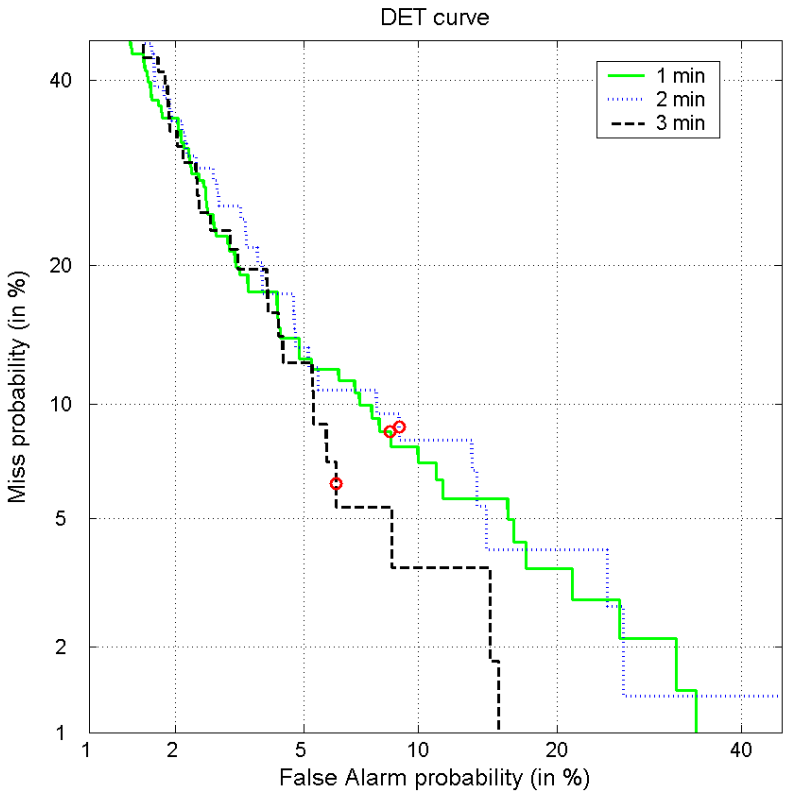

Figure 4.13: At least 2 speakers in common: Different conversation lengths using mean score. The EER point is marked with a red circle.

6.25%. In this case, about 67% of the false alarm errors are due to the conversations having exactly one speaker in common.

Once again we can adopt a semi-automatic approach to reduce the errors made by the system. For example, on set-III of the ENRON database, for a miss probability of 1.8%, the false alarm probability is 14.6%. Thus, the user need not check 85% of the data and needs to verify the remaining portion. If we can relax the miss probability to 4.5%, the false alarm probability is 8.6%. In this case, about 91% of the database need not be manually checked.

4.3.3 Sorting Conversations by Score

Each conversation in the database is used as a training conversation and the remaining conversations are tested against this conversation. A score is computed for every conversation pair. A higher score indicates that there is a greater probability that the conversations have a speaker in common. Thus, for each training conversation, we can sort (in a descending order) the remaining test conversations by the score. This sorted list of conversations can be very useful. A user can start from the top of the list and verify if the conversations have one or more speakers in common. The user can stop the verification once he or she encounters a certain number of consecutive false detections. The advantage of this semi-automatic technique is that the system does not make a final decision and hence does not require a decision threshold. Manual effort is also reduced as the files are sorted and most of the conversations do not have to be checked. In this manner, most of the

conversations having one or more speakers in common will be identified and errors of the system are manually corrected. The result of this approach will depend on the user's discretion and the number false detections that prompt the user to stop verifying the conversations.

This approach is illustrated for both the algorithms: (a) at least one speaker in common and (b) at least two speakers in common. Let N denote the number of conversations in the database. Figures 4.14 and 4.15 represent a N-1 x N matrix, where each column represents a training conversation and the testing files are sorted in each column with the files with higher score occupying the top rows. The conversations having one or more speakers in common are represented by a white square and the conversations with no speaker in common are represented with a gray square. It is desirable to have the white squares occupying the top rows since they have one or more speaker in common and should have a higher score.

Figure 4.14 shows the output of the system for at least one speaker in common for conversations that are at least three minutes in duration (N=112). We can observe that most of the white squares are clustered in the top rows and most of the gray area is in the lower part of the matrix. However, there are quite a few white squares occupying the lower rows of the matrix. These files are usually not retrieved and they contribute to the miss error rate. The gray squares in the upper rows contribute to the false alarm probability. The figure indicates that the user will be able to retrieve most of the conversations with one or more speakers in common (white squares) before the user encounters many false detections (gray squares).

Figure 4.15 shows the output of the system for at least two speakers in common

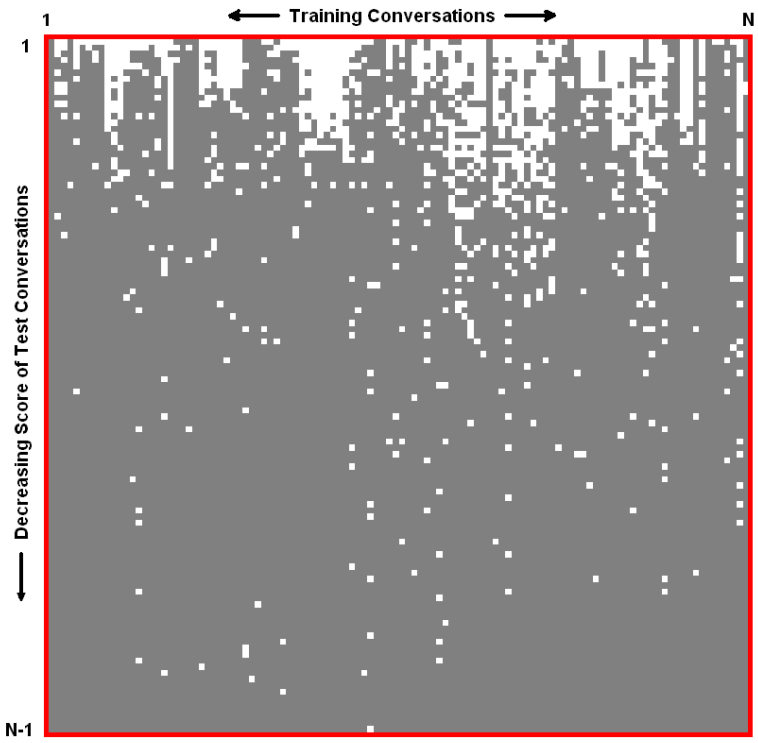

Figure 4.14: At least 1 speaker in common: Files sorted by score. White squares: files with at least 1 speaker in common. Gray squares: files with no speaker in common. Conversation length > 3 min.

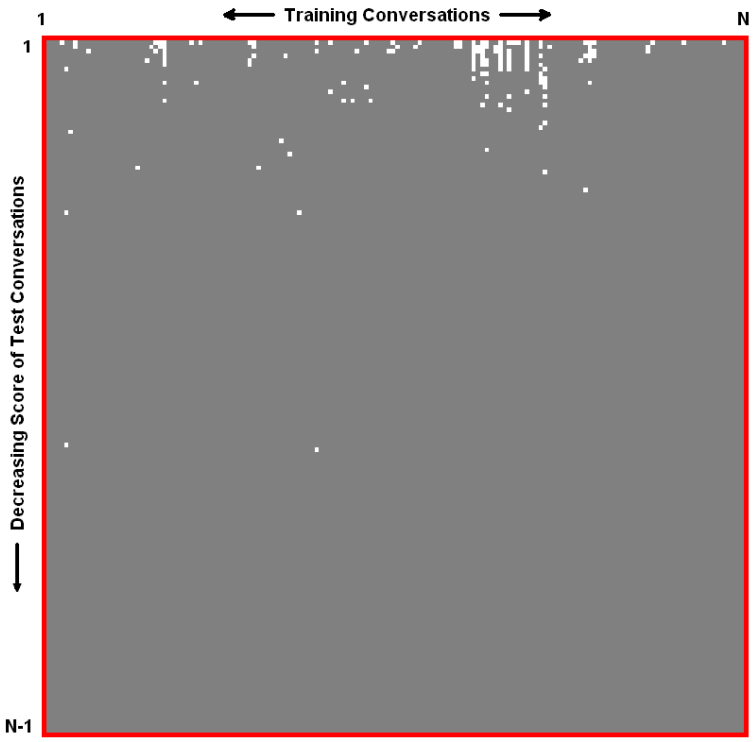

Figure 4.15: At least 2 speakers in common: files sorted by score. White squares: files with at least 2 speakers in common. Gray squares: files with no speaker in common. Conversation length > 2 min.

for conversations that are at least two minutes in duration (N=156). In this case, the performance of the system is better and most of the white squares are clustered at the top. The number of white squares in the lower part of the matrix is relatively smaller. The errors made by the system have been analyzed in the following section.

4.3.4 Error Analysis

The algorithm for at least one speaker in common makes more errors compared to the algorithm for at least two speakers in common. Here, we are referring to

the algorithms that do not use speaker segmentation. On the ENRON database, for conversations greater than three minutes in duration, the EER is 15.98% and 6.25% for at least one and two speakers in common, respectively. This difference in performance was also evident from Figures 4.14 and 4.15. The proportion of white squares occupying the lower rows of the matrix is greater for the case of at least one speaker in common. The algorithm for at least two speakers in common has superior performance since speaker detection accuracies increase with the length of the test utterance. For at least two speakers in common, the entire conversation is scored against the multi-speaker model. For the other algorithm, short segments of the test conversation are scored against the model. Some of the errors made by the two algorithms are discussed below.

4.3.4.1 At Least One Speaker in Common

First, we will concentrate on the second file (training file) in Set-III (minimum conversation length is three minutes) of the database. This conversation file contains a male speaker and a female speaker and is represented by the second column in Figure 4.16. The first three white squares at the top of the column(see zoomed area of file number 2) have the highest scores and these files have at least one speaker in common. The male speaker is common in these three files. The next two files have the fourth and fifth highest score. However, these files have no speaker in common with the second file and represent false detections. Both files contain a male and a female speaker. Further analysis shows that the segment in the 4^{th} and

5^{th} test files having the maximum score contains the female speaker. Hence, it is the female speakers of the two files that are more similar and contribute to a high score. Listening to these files reveal that the female speakers are indeed similar and there is a possibility of confusion.

It is the male speaker who is the common speaker for all the files represented by the white squares. Some of the files were analyzed and it was found that the segment yielding the maximum score does contain the common male speaker. The file with the 11^{th} highest score(white square) contains four speakers. Once again, the algorithm located the segment containing the male speaker, who is the common speaker. These segments usually capture the portions where the speaker is speaking continuously and contain single-speaker data. This is expected since the algorithm looks for the most positive evidence of the speaker.

Another file was chosen at random and analyzed. This is the 21^{st} file (training file) in the database and is represented by column number 21 in Figure 4.16 (also see the zoomed area for file number 21). The files having the highest and the third highest score are false alarms while the files having the second and fourth highest score are correct detections. The training file contains two male speakers and one of the male speakers in the files represented by white squares is common to the conversations. The segment with the maximum score contains the common speaker. The incorrect files were heard to determine if there are similar speakers present in the files. The file with the highest score does contain a very similar speaker and even after careful listening it is difficult to make a decision. In some of the other incorrect files the speakers are not very similar and it is relatively easier

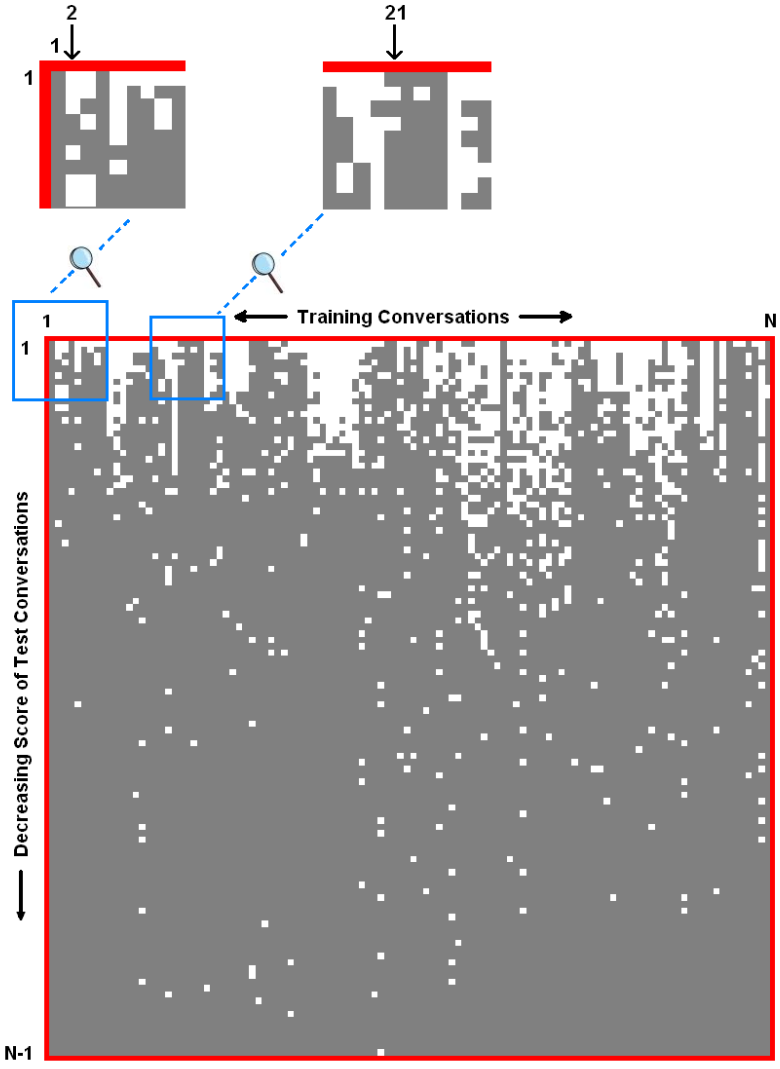

Figure 4.16: At least 1 speaker in common: Files sorted by score. White squares: files with at least 1 speaker in common. Gray squares: files with no speaker in common. Conversation length $>$ 3 min. The zoomed areas show the training files (file no. 2 and 21) that were analyzed.

to determine that they do not have a speaker in common. However, overall it is difficult to judge the degree of similarity of the speakers. Since these files mainly contain male speakers, it is difficult to discriminate between the speakers as their voices sound similar. Another problem, is that for some of the speakers the voice changes during the course of the conversation due to the speaker's emotion. Higher levels of information such as the speaking rate, accent, prosody and idiosyncracies (such as characteristic laughter) helped a lot in deciding if the conversations have a speaker in common. The linguistic content in the conversation, especially when the speakers mention their names, makes it easier to identify common speakers. The white squares at the bottom of the matrix represent the worst errors since these files received very low scores. These files were manually checked and they do have a speaker in common. The reason behind this error is not entirely clear. It is probably due to biased conversations or considerable mismatch in the channels.

4.3.4.2 At Least Two Speakers in Common

First, we will consider the fifth file (training file) in Set-II (minimum conversation length is two minutes) of the database. This file contains two male speakers and is represented by the fifth column in Figure 4.17 (also see the zoomed area for file number 5). The algorithm does not work very well for this file and the files with at least two speakers in common have a rank of 7, 39 and 91 in the column containing 155 files. The top three files have a high score because they have one speaker in common. The speakers in the file with a rank of four are similar to the

speakers in the file we are comparing with. However, the speakers in the files with a rank of 5 and 6 are different and it is easy for a human to judge that the system has made an error. Even though all the speakers in these files are male, the differences in their voice qualities made it easier to discriminate between the speakers. The files that have two speakers in common also contain a third speaker that is not present in the training file. This additional speaker can be the cause of the low score obtained in these cases. The two white squares towards the bottom of the matrix represent the worst errors of the system. These two white squares appear due to a conversation pair that have two male speakers in common and one file has another female speaker as well. In this, case the presence of the additional speaker adversely affects the algorithm. This effect will probably be reduced if the additional speaker were of the same gender.

The seventh file in the database contains one male and one female speaker. The other two conversations in the database containing both the speakers correctly receive the highest and second highest score (see zoomed area for file number 7 in Figure 4.17). It is interesting to note that the next ten files contain at least one male and at least one female speaker. Some of the files also contain the common male speaker.

Another matrix is shown to analyze the output of the algorithm for at least two speakers in common (see Figure 4.18). There are three types of squares in the figure: the white squares represent the files that have at least two speakers in common, the gray squares represent the files that have exactly one speaker in common and the black squares represent the files with no speaker in common. The figure shows that

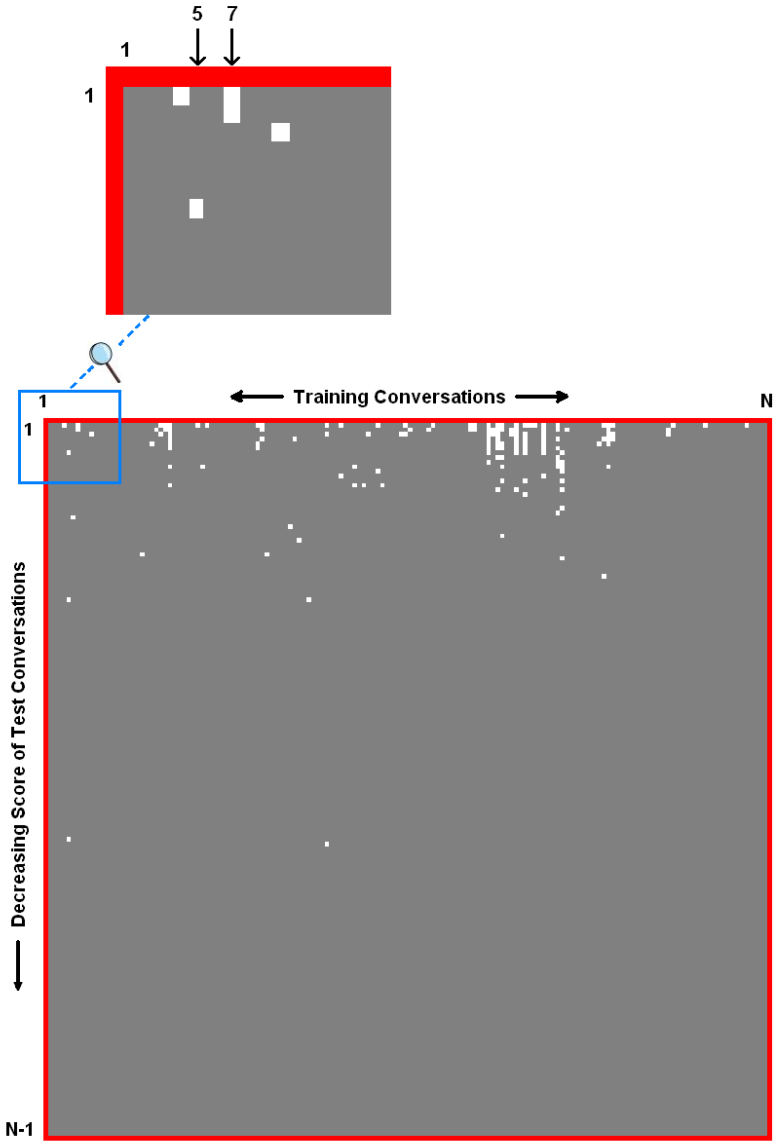

Figure 4.17: At least 2 speakers in common: files sorted by score. White squares: files with at least 2 speakers in common. Gray squares: files with no speaker in common. Conversation length > 2 min. The zoomed areas show the training files (file no. 5 and 7) that were analyzed.

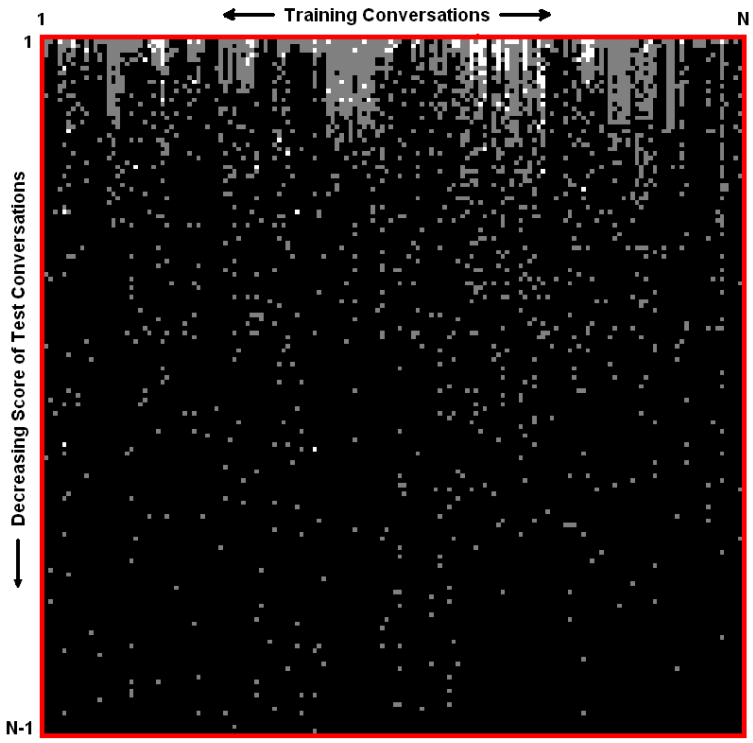

Figure 4.18: At least 2 speakers in common: files sorted by score. White squares: files with at least 2 speakers in common. Gray squares: files with exactly one speaker in common. Black squares: files with no speaker in common. Conversation length > 2 min

that files with exactly one speaker in common (gray squares) usually receive a higher score compared to the files with no speakers in common (black squares). This is expected since the presence of the common speaker improves the score of the entire conversation. In fact, sometimes the score is so high that these files have a higher rank than the files with at least two speakers in common (white squares). Overall, the algorithm for at least two speakers in common has lesser number of errors as compared to the algorithm for at least one speaker in common.

4.3.5 With Manual Speaker Segmentation

The goal of this experiment is to determine if speaker detection is improved by using manual speaker segmentation as opposed to the automatic techniques. Manual segmentation yields single-speaker speech data, which can be used for training a target speaker model. Ten target speakers were considered. The ten speakers selected are present in a number of conversations so that the number of target trials is increased. Manual segmentation was performed on a conversation till we have two minutes of speech of the target speaker. For each target speaker, two conversations were manually segmented yielding four minutes of speech for training the model. Note, that manual segmentation is only used to create training data and the test files are still unsegmented conversations. Speaker detection was then performed on these test files using the target model.

4.3.5.1 With and Without Speaker Segmentation of Test File

There are two ways to perform speaker detection in a conversation. Automatic speaker segmentation can be used to divide the test file into single-speaker portions and then speaker verification can be used. The maximum score of all the single-speaker portions is the detection score. In the other approach, the test file is divided into overlapping segments and each segment is scored against the speaker model. The maximum score over all the segments is the detection score. Two minutes of speech was used to train the target speaker model. A 1024-mixture UBM was used and the conversations over two minutes were evaluated.

The DET curves of the two approaches are overlapping and they cross each other (Figure 4.19). The method using speaker segmentation has a lower EER and performs better in the region of low miss probability. The EER with automatic speaker segmentation of test file is 20.97% and without speaker segmentation is 21.80%. However, even after using manual speaker segmentation to train a target speaker model, the EER is high. In the next experiment, the amount of training data is doubled.

4.3.5.2 Effect of Training Data

In this case, two conversations are used to train the target speaker model, yielding four minutes of training data. Increasing the amount of training data improves the speaker model leading to a dramatic inprovemnt in the performance. A 1024-mixture GMM was used automatic speaker segmentation was employed. The

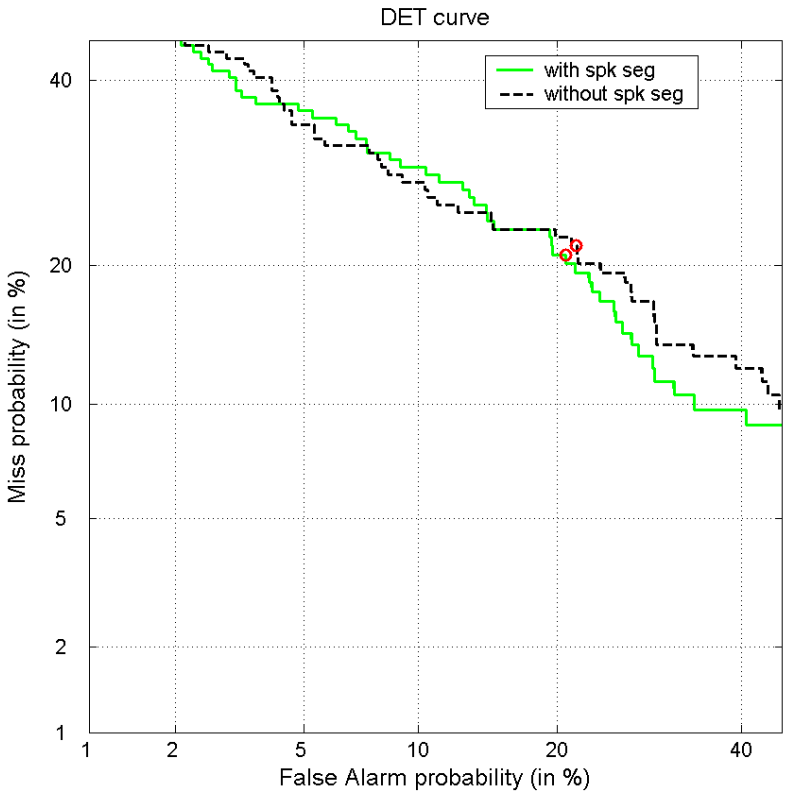

Figure 4.19: 1 speaker detection: With and without speaker segmentation of the test file. Manual segmentation was used to create training data. Conversation length > 2 min. The EER point is marked with a red circle.

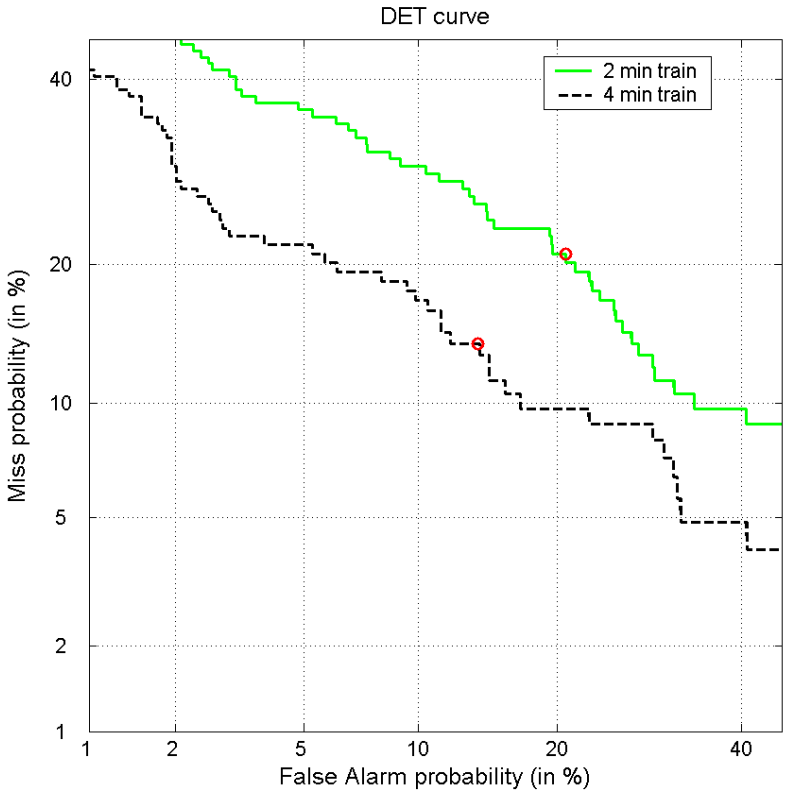

Figure 4.20: 1 speaker detection: Manual segmentation was used to create training data with training lengths of 2 mins and 4 mins. Conversation length > 2 min. The EER point is marked with a red circle.

EER for two minutes of training data is 20.97% and for four minutes it is 13.72% (Figure 4.20).

4.3.5.3 Comparison of Manual Segmentation, Automatic Segmentation and No Segmentation

In this experiment a comparison of the automatic approaches and the manual segmentation technique was done. The comparison is not exact since manual segmentation allows us to perform one speaker detection using the target model. When

using automatic approaches, we are doing at least one speaker detection. Hence the number of positive instances is greater in this case. The number of target trials for one speaker detection using manual segmentation is 124 and the number of impostor trials is 1436.

The results are summarized in Table 4.15. Figure 4.21 shows the three DET curves. The target models trained on four minutes of speech clearly outperform the automatic methods. However, manual segmentation requires time and effort. In this case, the approach with speaker segmentation does slightly better than the method without speaker segmentation. However, in all the other experiments, the approach without speaker segmentation outperforms the algorithm with speaker segmentation.

Table 4.15: Equal error rate for different algorithms

Algorithm	**Equal Error Rate (EER)**
No speaker segmentation	24.26
Automatic speaker segmentation	22.45
Manual speaker segmentation	13.72

Thus, there are various techniques that can be used for voice mining of such a database. There is a tradeoff between the amount of manual effort required and the performance of the system. The threshold of the system can be varied to give different miss error rates and false alarm error rates. The decision regarding what the threshold should be will depend on the task at hand. Since the automatic

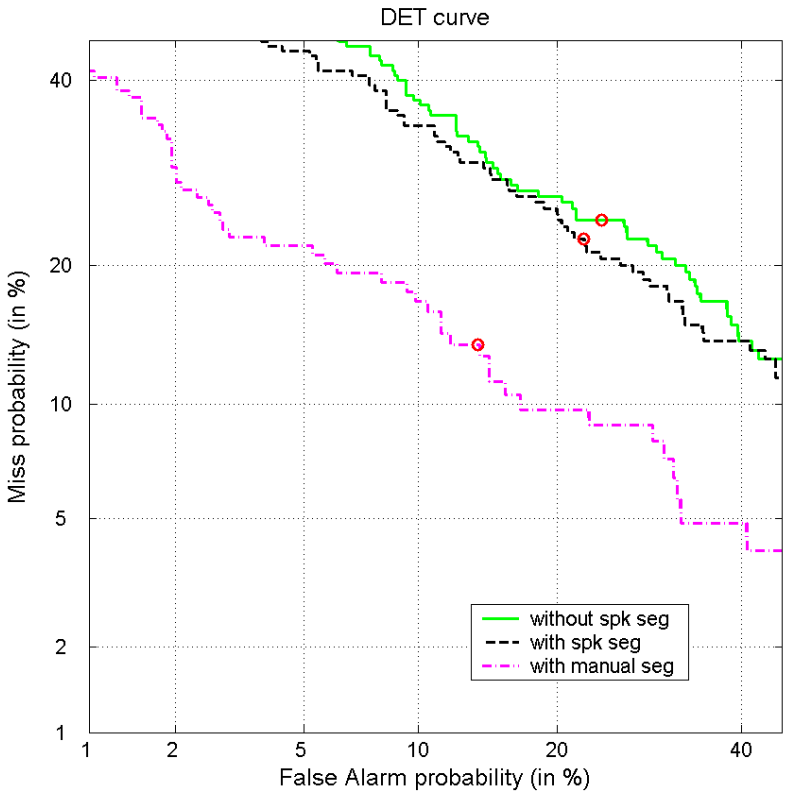

Figure 4.21: At least 1 speaker in common: manual segmentation, with speaker segment and without speaker segmentation. Conversation length > 2 min different train length. The EER point is marked with a red circle.

techniques are not free of errors, we need to know which type of error is less costly for the task and set the threshold accordingly. It might be better to adopt a semi-automatic approach where the system helps the user in the voice mining of the speech database. In this way, the human effort involved is reduced considerably and the errors made by the system are corrected.

Chapter 5

Conclusions and Future Work

In this study, a new voice mining scenario was considered. Given a database of telephone conversations, the task is to find conversations that have one or more speakers in common. This is different from normal speaker detection tasks, where a prior target model is available and the task is to find the speaker in multi-speaker speech files. However, in our case no prior target models are available and there is no demarcation between training and testing data. Another issue that had to be considered was the poor audio quality of the database. Usual speaker detection approaches involve speaker segmentation of the multi-speaker conversations. However, the speaker segmentation algorithm does not work very well due to poor audio quality and that the number of speakers in each conversation is unknown. A new technique was developed which does not require speaker segmentation by training a multi-speaker model on the entire conversation. Each conversation is used to train a multi-speaker model and speaker detection is performed on the remaining conversations in the database. Various scoring strategies were tried and were compared to the speaker segmentation approach. The experiments demonstrate that this technique outperforms the speaker segmentation approach and a combination of the two approaches can be used to achieve improved performance.

The automatic identification of conversations that have one or more speakers

in common establishes links between the conversations and helps in the browsing of such a database. A user can select a particular conversation that is of interest and the system automatically retrieves all the conversations that have at least one or at least two speakers in common. However, the system is not perfect and can be adjusted to minimize false rejections at the cost of false detections. The false detections can be discarded by manually checking the retrieved conversations. The advantage of using such a system is that the manual effort is considerably reduced as the person has to check a small number of files compared to the number of files in the database. The system can help the user to browse the entire database and establish all the links between the conversations.

Another semi-automatic approach for the mining of such a database may involve manual speaker segmentation. Manual segmentation of the conversations can be done on a portion of the database. The single-speaker data can be used to construct target speaker models and they can be used for speaker detection. The advantage of this manual segmentation is that we can train good speaker models and the performance of the system can be improved. However, the disadvantage is that the segmentation task is time consuming and it requires considerable manual effort. A large number of conversations need to be segmented if the number of speakers in the database is high.

The results indicate that even while using manual segmentation (as done on the ENRON database) or perfect separation of the speakers (as done on the Switchboard-I database) the system is still not free from errors. The performance of the speaker detection system can be improved by using better features that capture

speaker-specific characteristics. There is a need to find more robust features that are not adversely affected by channel variations and noise. There is ongoing research both in the area of low-level features and high-level features. A set of low-level acoustic parameters have been developed that capture both the speaker's source information and vocal tract information. These parameters were successfully used for the text-independent speaker identification task and the details can be found in [6]. High-level features can also be useful in characterizing speaker-specific traits and speaking habits and can help improve performance [3].

Future work can also involve the use of a speech recognition engine for name-spotting or word-spotting in the telephone conversations. This information can be useful in determining the speakers in the conversations and improving detection performances. Instead of using a single Gaussian Mixture Model to characterize a speaker, a supervised approach can be adopted. This would involve extracting different features depending on the sound classes in the speech and building models for the various sound classes of a speaker. Word-spotting would also allow the construction of speaker-specific word level models and Hidden Markov Models can be used for this purpose. These different levels of information and speaker modeling approaches can be used to improve speaker detection performance.

BIBLIOGRAPHY

[1] D.A. Reynolds. An overview of automatic speaker recognition technology. In *Proceedings of ICASSP*, volume 4, 2002.

[2] G.R. Doddington, M.A. Przybycki, A.F. Martin, and D.A. Reynolds. The NIST speaker recognition evaluation-overview, methodology, systems, results, perspective. *Speech Communication*, 2000.

[3] D. Reynolds, W. Andrews, J. Campbell, J. Navratil, B. Peskin, A. Adami, Q. Jin, D. Klusacek, R. Mihaescu, J. Godfrey, D. Jones, and B. Xiang. The supersid project: Exploiting high-level information for high-performance speaker recognition. Technical report, 2003.

[4] J.P. Campbell. Speaker recognition: a tutorial. In *Proceedings of the IEEE*, volume 85, pages 1437–1462, September 1997.

[5] M. Plumpe, T. T.Quatieri, and D. Reynolds. Modeling of the glottal flow derivative waveform with application to speaker identification. *IEEE Transactions on Speech and Audio Processing*, 1(5):569–586, September 1999.

[6] C.Y. Espy-Wilson, S. Manocha, and S. Vishnubhotla. A new set of features for text-independent speaker identification. In *Proceedings of ICSLP*, 2006.

[7] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of IEEE*, volume 77, pages 258–286, February 1989.

[8] D.A. Reynolds and R. Rose. Robust text-independent speaker identification using gaussian mixture speaker models. *IEEE Transactions on Speech and Audio Processing*, 3, 1995.

[9] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki. The det curve in assessment of detection task performance. In *Proceeding of EuroSpeech*, volume 4, pages 1895–1898, 1997.

[10] D.A. Reynolds, T.F. Quatieri, and R.B. Dunn. Speaker verification using adapted gaussian mixture models. *Digital Signal Processing*, 2000.

[11] R. Gazit and Y. Metzger. Voice mining with multiple target speakers. In *Proceedings of the Odyssey Workshop*, 2004.

[12] M. Przybocki and A. Martin. NIST speaker recognition evaluation chronicles. In *Proceedings of the Odyssey Workshop*, 2004.

[13] http://www.nist.gov/speech/tests/spk/.

[14] R. Dunn, D.A. Reynolds, and T. Quatieri. Approaches to speaker detection and tracking in conversational speech. *Digital Signal Processing Review Journal*, January 2000.

[15] D. Gillick, B. Peskin, and S Stafford. Speaker detection without models. In *Proceedings of ICASSP*, volume 1, 2005.

[16] J.F. Bonastre, S. Meignier, and T. Merlin. Speaker detection using multi-speaker audio files for both enrollment and test. In *Proceedings of ICASSP*, volume 2, 2003.

[17] J. McLaughlin and D.A. Reynolds. Speaker detection and tracking for telephone transactions. In *Proceedings of ICASSP*, volume 1, 2002.

[18] http://jikd-email.umiacs.umd.edu/.

[19] http://www.enrontapes.com/.

[20] P. Delacourt and C.J. Wellekens. DISTBIC: A speaker-based segmentation for audio indexing. *Speech Communication*, 32, September 2000.

[21] J. Godfrey, C. Holliman, and J. McDaniel. Switchboard: Telephone speech corpus for research and development. In *Proceedings of ICASSP*, 1992.

[22] http://www.ldc.upenn.edu/.

[23] J. Pelecanos, U. Chaudhari, and G. Ramaswamy. Compensation of utterance length for speaker verification. In *Proceedings of the Odyssey Workshop*, 2004.

[24] H. Hermansky and N. Morgan. Rasta processing of speech. *IEEE Transactions on Speech and Audio Processing*, 2(4):578–589, October 1994.

[25] J. Pelecanos and S. Sridharan. Feature warping for robust speaker verification. In *Proceedings of the Odyssey Workshop*, 2001.

[26] C. Bron and J. Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16:575–577, 1973.