Tools for Automatic Audio Indexing

Kasper Winther Jørgensen and Lasse Lohilahti Mølgaard

Kongens Lyngby 2006
IMM-THESIS-2006-31

Abstract

Current web search engines generally do not enable searches into audio files. Informative metadata would allow searches into audio files, but producing such metadata is a tedious manual task. Tools for automatic production of metadata are therefore needed. This project investigates methods for audio segmentation and speech recognition, which can be used for this metadata extraction.

Classification models for classifying speech and music are investigated. A feature set consisting of zero-crossing rate, short time energy, spectrum flux, and mel frequency cepstral coefficients is integrated over a 1 second window to yield a 60-dimensional feature vector. A number of classifiers are compared including artificial neural networks and a linear discriminant. The results obtained using the linear discriminant are comparable with the performance of more complex classifiers. The dimensionality of the feature vectors is decreased from 60 to 14 features using a selection scheme based on the linear discriminant. The resulting model using 14 features with the linear discriminant yields a test misclassification of 2.2%.

A speaker change detection algorithm based on a vector quantization distortion (VQD) measure is proposed. The algorithm works in two steps. The first step finds potential change-points and the second step compensates for the false alarms produced by the first step. The VQD metric is compared with two other frequently used metrics: Kullback Leibler divergence (KL2) and Divergence Shape Distance (DSD) and found to yield better results. An overall F-measure of 85.4% is found. The false alarm compensation shows a relative improvement in precision of 59.7% with a relative loss of 7.2% in recall in the found change-points. The choice of parameters based on one data set generalize well to other independent data sets.

The open source speech recognition system SPHINX-4 is used to produce transcripts of the speech segments. The system shows an overall word accuracy of $\sim 75\%$.

Resumé

Nuværende web-søgemaskiner muliggør ikke direkte søgning i lydfiler. Informativt metadata kunne gøre det muligt at søge i lydfiler, men fremstilling af metadata er en langsommelig opgave, hvilket taler for at anvende automatiske metoder til at udtrække disse data. Dette projekt undersøger metoder til segmentering af lyd og talegenkendelse, hvilket kan bruges til at producere de omtalte metadata.

Klassifikationsmodeller til klassificering af tale og musik undersøges. Ved at integrere et sæt 'features', bestående af nulpunktskrydsninger, kort-tids energi, spektrum flux og mel kepstrale koefficienter, over et sekund konstrueres et 60-dimensionalt feature-sæt. Et antal klassificeringsmodeller undersøges, heriblandt kunstige neurale netværk og lineære diskriminanter. Resultaterne viser at den lineære diskriminant opnår lige så gode resultater som mere komplicerede klassificeringsmodeller. Dimensionaliteten af feature sættet bliver formindsket fra 60 til 14 features ved at benytte en udvælgelsesmetode baseret på den lineære diskriminant. Den endelige model, der benytter 14 features med den lineære model, opnår en fejlklassifikation på 2,2%.

En talerskift-detekterings-algoritme baseret på et 'vector quantization distortion' (VQD)-mål bliver foreslået. Algoritmen er opdelt i to trin. Første trin finder potentielle talerskift og andet trin kompenserer for de falske positiver, der bliver introduceret i første trin. VQD-målet bliver sammenlignet med de to andre ofte brugte mål: Kullback-Leibler divergens (KL2) og 'Divergence Shape Distance' (DSD). VQD overgår de to andre. Forsøgene med VQD giver et F-mål på 85.4%. Metoden til at mindske antallet af falske positiver giver en relativ forbedring i præcision på 59.7% med et fald i antallet af fundne taler-skift på 7.2% relativt. Parametrene valgt ud fra ét data sæt generaliserer godt til andre uafhængige sæt.

Open-source talegenkendelsessystemet SPHINX-4 bliver brugt til at generere transskriptioner af talesegmenterne. Systemet giver en overordnet ord-nøjagtighed på $\sim 75\%$.

Preface

This Master's thesis was carried out at the Technical University of Denmark, Department of Informatics and Mathematical Modelling, in the period from September 2005 to March 2006, under the supervision of Professor Lars Kai Hansen.

Parts of the thesis are to appear in the paper UNSUPERVISED SPEAKER CHANGE DETECTION FOR BROADCAST NEWS SEGMENTATION, which has been submitted to the European Signal Processing Conference (EUSIPCO) 2006. The paper can be found in appendix A.

Lasse Lohilahti Mølgaard
s001514

Kasper Winther Jørgensen
s001498

Lyngby, 15th March 2006

Contents

List of Figures

List of Tables

Nomenclature

Symbols

α_{cd}	Amplifier for $\mathrm{th}_{\mathrm{cd}}$
α_{fac}	Amplifier for $\mathrm{th}_{\mathrm{fac}}$
$A = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$	Feature vector sequence
$B = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$	Feature vector sequence
C_k	Class k
$\mathbf{C}$	Vector quantization code-book
$\mathbf{c}_j$	Vector quantization code-vector
$d_E(\mathbf{x}, \mathbf{y})$	Euclidean distance between two feature vectors
l_{aw}	Length of analysis windows
l_s	Shift in change-point detection algorithm
$\mathbf{t} = (t_1, t_2, ..., t_c)$	Target-vector
$T_{\max}$	Maximum window length
T_i	Interval time for peak detection
$\mathbf{x} = (x_1, x_2, ..., x_d)$	Input-vector
w_i	word i

Abbreviations

ASR	Automatic Speech Recognition
CD	Change Detection
CMN	Cepstral Mean Normalization
DFT	Discrete Fourier Transform
DSD	Divergence Shape Distance
F	F-measure
FAC	False Alarm Compensation
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
HZCRR	High Zero-Crossing Rate Ratio
IDFT	Inverse Discrete Fourier Transform
KL2	Symmetric Kullback-Leibler distance
K-means	K-Means clustering algorithm
LM	Language Model
K-NN	K-Nearest Neighbors
LPC	Linear Prediction Coefficients
LSTER	Low Short Time Energy Ratio
LVASR	Large Vocabulary Automatic Speech Recognition
MFCC	Mel Frequency Cepstral Coefficients
ΔMFCC	Delta Mel Frequency Cepstral Coefficients
$\Delta\Delta$MFCC	Delta Delta Mel Frequency Cepstral Coefficients
NN	Neural Network
PCA	Principal Component Analysis
PRC	Precision
RCL	Recall
STE	Short Time Energy
SF	Spectrum Flux
$\mathrm{th}_{\mathrm{cd}}$	Threshold used in change-point detection
$\mathrm{th}_{\mathrm{fac}}$	Threshold used in false alarm compensation
VQ	Vector Quantization
VQD	Vector Quantization Distortion
WA	Word Accuracy
WER	Word Error Rate
ZCR	Zero-Crossing Rate

Chapter 1

Introduction

The amount of audio available in different databases on the Internet today is immense. Successful access to such large amounts of data requires efficient search engines. Traditional web search engines, such as Google, are often limited to text and image indexing, thus many multimedia documents, video and audio, are excluded from these classical retrieval systems. Even systems that do allow searches for multimedia content, like AltaVista and Lycos, only allow queries based on the multimedia filename, nearby text on the web page containing the file, and metadata embedded in the file such as title and author. This might yield some useful results if the metadata provided by the distributor is extensive. Producing this data is a tedious manual task, and therefore automatic means for creating this information is needed.

Today many radio stations provide entire news or talk shows in form of podcasts or streaming services, with general headlines of the content. In general no detailed index of the file is provided, which makes it time consuming to look for the part of interest.

The optimal division of radio shows, such as broadcast news, debate programs, and music programs, should be based on the topics covered in each part. Such a division requires that it is possible extract topics and the parts related to this topic. Topic detection requires transcription of the speech parts of the audio, but adding audio cues would aid in retrieving coherent segments.

Adding audio cues is done by segmenting the audio based on the characteristics of the audio stream. The segments generated can then be summarized on basis of the type of audio.

- Music clips would be described by genre and artist.
- Speech summaries naturally and would consist of identities of speakers and a transcription of what is said.

Below we will present some of the efforts that have been done to access audio files and spoken documents in particular.

1.1 Audio Retrieval Systems

Research in audio retrieval has mainly been focused on music and speech. Music retrieval has focused on quantifying different characteristics of the music such as mood, beat, and other characteristics to classify genre or find similarities between songs. This area is not the focus of this thesis and will not be covered further.

Approaches to spoken document retrieval have included automatic broadcast news transcription and other speech retrieval systems. These system are described below.

1.1.1 Automatic Broadcast News Transcription

Broadcast news transcription has been heavily researched and is considered to be the most demanding assignment in speech recognition, as the speakers and conditions vary much in the course of a news show. The speakers range from anchor speaking in an ideal environment to reporters speaking from noisy environments on an non-ideal telephone line. Non-native English speakers make the problem even more challenging. The shows often use music between different parts of the show and in the background of speakers.

These systems therefore often include a segmentation preprocessor, that removes the non-speech parts and finds segments with homogenous acoustical characteristics. In this way the speech recognizers can be adjusted to the differing acoustical environments. Solving the task has required advances in both preprocessing and speech decoding.

The performance of these systems are evaluated in annual Rich Transcription workshops arranged by [NIST, 2005]. The participants include commercial groups such as IBM and BBN and academical groups such as LIMSI and CU-HTK, whose systems are described in [Gauvain et al., 2002] and [Johnson et al., 2001], respectively. The aim of these projects is not directly audio indexing, but to transcribe the spoken words as precisely as possible. As mentioned above the systems incorporate audio classification to identify acoustically similar parts of the audio. The classification is done so the recognition engines can be optimized for female/male speakers, or narrow-/wideband channels, i.e. telephone or studio speech.

1.1.2 Speech Retrieval Systems

More retrieval oriented projects have also been developed in recent years. One approach to access audio data (more or less derived from the broadcast news transcription efforts) is presented in [Van Thong et al., 2002] namely the Speechbot-service. The system was designed to index and transcribe a number of radio shows streamed from the Internet.

The Speechbot-service is not using segmentation based on characteristics of the speech stream, which means that the results are very long transcripts without any indication of speaker turns. Instead the transcripts are indexed using a text-indexing engine to facilitate user queries. The result of a query is the place in the transcript were the match is found. For users to hear the part of the radio show returned by the query that is found, the group has implemented a way to align the very long transcripts with the audio by assigning time marks for every word very accurately. This service has only been running on an experimental basis and was closed at the beginning of November 2005.

The SpeechFind project [Hansen et al., 2005] is a recent project that takes up speech retrieval on a very large scale. The project is a part of the National Gallery of the Spoken Word that contains speech recordings going back to the 19th century[1]. The project includes research in metadata collection, audio segmentation, and speech recognition to produce segmented transcripts that are searchable. Because of the very large scale of the database (as much as 60,000 hours of data) efficiency of the algorithms and automating the indexing are focus-points of the research. The sound quality of the recordings varies greatly. Therefore another important topic for this project is to improve noise robustness of speech recognition and apply different speech analysis techniques such as accent recognition.

1.2 Segmentation Approaches

The segmentation approaches used in the systems mentioned above and in other retrieval systems cover a wide range of methods. In general the methods can be divided into two groups: audio classification and change detection. These approaches have different attributes that qualify them for different uses as presented below.

1.2.1 Audio Classification

As mentioned above the typical first part of a speech retrieval system concerns identifying different audio classes. The four main classes considered are speech, music, noise, and silence but depending on the application more specific classes such as noisy speech, speech over music, and different classes of noise, have been considered.

The task of segmenting or classifying audio into different classes has been implemented using a number of different schemes. Following the paper by [Saunders, 1996] a multitude of approaches have been proposed. Two aspects that must be considered are feature and classification model selection.

Different features have been proposed, based on different observations on the characteristics that separate speech, music and other possible classes of audio. The features are generally divided on basis of the time horizon they are extracted.

The simplest features proposed include time domain and spectral features. Time domain

[1]The recordings include some of Edison's first cylinder disks ranging over famous recordings such as "A small step for man..." to recent presidential speeches

features typically represent a measure of the energy or zero crossing counts.

Cepstral coefficients have been used with great success in speech recognition systems, and subsequently have shown to be quite successful in audio classification tasks as well [Gauvain et al., 2002]. Other features have also been proposed based on psychoacoustic observations, see e.g., [McKinney and Breebaart, 2003].

The other aspect to be considered is the classification scheme to use. A number of classification approaches have been proposed, that can be be divided into rule-based and model-based schemes. The rule-based approaches use some simple rules deducted from the properties of the features. As these methods depend on thresholds, they are not very robust to changing conditions, but may be feasible for real-time implementations.

Model-based approaches have included Maximum A Posteriori (MAP) classifiers, Gaussian Mixture Model (GMM), K-nearest-neighbor (K-NN), and linear perceptrons. Another approach in this context is to model the time sequence of features, or the probability of switching between different classes. Hidden Markov Models (HMM) take this into account.

1.2.2 Speaker Change Detection

Another approach to identify homogenous audio segments could be done by performing event detection.

Indexing the speech segments produced by audio classification into homogenous speaker segments is a natural part of an audio retrieval system. The indexing task has an impact on the performance of speaker dependent speech recognizers. This will allow the system to adapt its acoustic model to the given speaker and/or environment and thereby improve recognition performance. Indexing speech streams into speaker turns also eases the readability of the transcriptions made by an automatic transcription system.

Approaches to speaker change detection can be divided into supervised and unsupervised methods. If the number of speakers and identities are known in advance, supervised models for each speaker can be trained, and the audio stream can be classified accordingly. If the identities of the speakers are not known in advance unsupervised methods must be employed.

Unsupervised speaker change detection approaches can roughly be divided into two classes, namely energy-based and metric-based:

Energy-based methods rely on thresholds in the audio signal energy. Changes are found at silence-periods. In broadcast news the audio production can be quite aggressive, with only little if any silence between speakers, which makes this approach less attractive.

Metric-based methods basically measure the difference between two consecutive frames that are shifted along the audio signal, with some minor variations. A number of distance measures have been investigated based on different parametric methods.

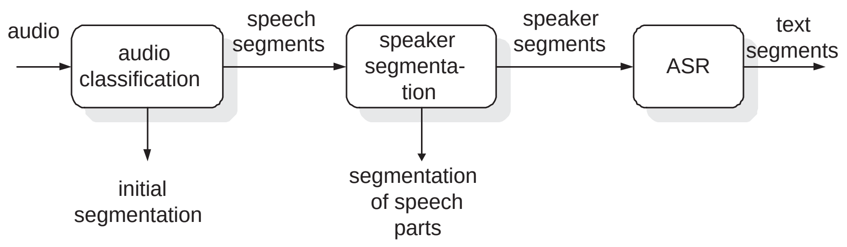


Figure 1.1: Overview of the system developed in this project. The audio is first classified to obtain segments containing either music or speech. The speech parts can then be segmented based on speaker changes. Finally the speech is transcribed, for each speaker segment.

A final approach proposed by [Kemp et al., 2000, Kim et al., 2005] is to develop hybrid-methods that use unsupervised methods to make an initial partitioning. Subsequently these segments are used to make models for re-segmentation.

1.2.3 Speech Recognition

Transcription of the segmented speech is done using an automatic speech recognition system. Large vocabulary, speaker independent speech recognition has been widely researched in the last decades and many freely available systems have been developed.

The efforts in broadcast news transcription have addressed speech recognition in noisy and other non-ideal environments. The results from these efforts can therefore be used in our context.

Large vocabulary speech recognition systems are typically implemented using Hidden Markov Model-based decoding but refinements and optimizations to these schemes are developed continuously.

1.3 Project Description

The objective of this project is to make a system that can provide metadata for audio documents. In this way the audio documents can be searched using current search engines. The project will focus especially on speech. Spoken documents often deal with numerous different topics. To aid in finding well-defined topic boundaries, different audio cues can be used. The aim of this project is to investigate initial methods for this task.

The project will deal with three distinct methods for producing metadata for audio documents. The parts of the system are shown in figure 1.1.

The first part of the system will perform audio classification. Audio classification is used to perform an initial segmentation especially focused on extracting speech from the audio stream. This aim of this classification is to split the audio into speech and music segments. We will especially investigate which features should be used and how they should be represented for the classification task. Given a number of features we will look into a number of different classifiers.

The next part concerns segmentation of speech into segments containing one speaker. In addition to provide an useful indexing, this should also improve the performance of the speech recognition and help the readability of the produced transcription. The change detection part will look into metric based methods used for this domain.

The last part of the system will obtain transcriptions of the speech segments using an automatic speech recognition system. As mentioned above, numerous speech recognition systems have been developed, and the implementation is a comprehensive task. Therefore an existing system will be employed.

The project will focus on broadcast news shows. The system should be able to generalize to a broad number of channels, i.e., the system will not be adapted to a single radio station or employ any kind of assumptions on speaker identities.

1.4 Report Overview

Our investigation of the problem is split into three parts, resembling the organization in figure 1.1. The three parts will be investigated, implemented, and evaluated separately. Because of this separation each chapter will be introduced with a presentation of previous work in the corresponding field. Initially though, we will introduce the features as they are common for the three parts. Finally the system will be combined to show the application on an example. The chapters include:

Chapter 2 - Feature Extraction:
: The feature extraction will be covered collectively in this chapter as the features are reused in the following three tasks.

Chapter 3 - Audio Classification:
: Presents audio classification. Features are evaluated for the task and a number of classifiers are tested to find an appropriate setup.

Chapter 4 - Speaker Change Detection:
: Presents the methods used to segment the speech parts to find speaker changes in the audio stream.

Chapter 5 - Speech Recognition:
: This chapter presents state-of-the-art speech recognition. Speech recognition engines in general and the system chosen for this system is described, and the setup of the system is reviewed.

Chapter 6 - Full System Example:
: Presents the performance of the combined system using an example.

Chapter 7 - Conclusion:
Gives the conclusion and further work.

Chapter 2

Feature Extraction

Feature extraction is a very important issue to get optimal results in our application. Extracting the right information from the audio increases the performance of the system and decrease the complexity of subsequent algorithms.

Generally applications require different features enhancing the characteristics of the problem. The features considered in this project are zero-crossing rate, short-time energy, spectrum flux, and mel-weighted cepstral coefficients. These features will be used for the audio classification part. For the speaker change detection and speech recognition only the mel-weighted cepstral coefficients will be used.

In this chapter the features will be described and illustrated through an example, where the features are extracted from speech and music samples.

2.1 Audio Preprocessing

The type of signals we are dealing with, namely speech and music, are so called quasi-stationary signals. This means that they are fairly stationary over a short period of time. This encourages the use of features extracted over a short time period.

In this project the audio used is in 16 kHz, 16 bit, PCM wave format. The audio is partitioned into frames of 20ms, with an overlap of 10ms. This means that each second of audio contains 100 frames, and each frame contains 320 samples.

A well-known problem of framing, comes from the truncation of the signal. The discontinuity induced by using a rectangular window can be reduced by using a windowing function with a greater attenuation in the side lobes. The Hamming and Hann windows accomplish this as

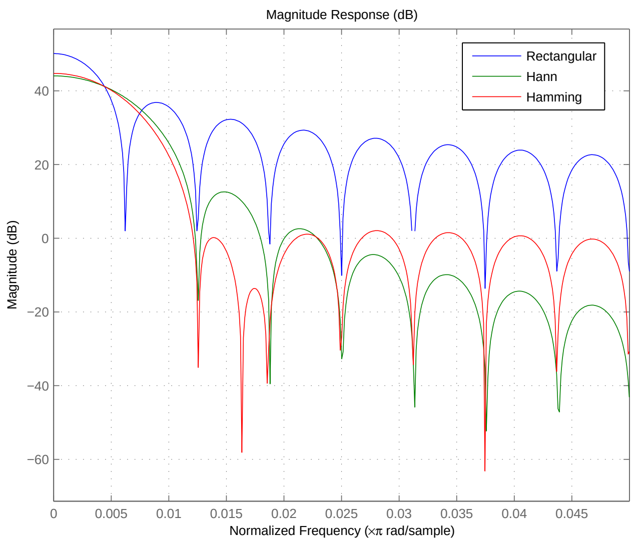


Figure 2.1: Attenuation of the lobes using different windowing functions (rectangular, Hann, Hamming)

seen in figure 2.1. As seen the Hamming window gives a better attenuation in the secondary lobes. The Hamming and Hann windows are from the family of raised cosine windows given by:

$$w(n) = \psi - (1-\psi)\cos\left(\frac{2\pi n}{N-1}\right) \tag{2.1}$$

Where for a Hamming window $\psi = 0.54$ and for Hann $\psi = 0.50$. N is the length of the signal to be filtered.

In this project we will use the Hamming filter, thus let $s_i(n)$ denote the i'th frame of the original audio signal. Then the Hamming filtered frame is given by:

$$x_i(n) = w(n)s_i(n) \tag{2.2}$$

2.2 Zero-Crossing Rate

Zero-crossing rate (ZCR) has been used in a wide range of audio applications. It has especially been used in audio classification and segmentation applications, see e.g., [Lu and Zhang, 2005, Huang and Hansen, 2004b].

ZCR is defined as the number of sign changes in a frame divided by the length N of the frame:

$$\text{ZCR} = \frac{1}{N}\sum_{n=2}^{N}\Big|\text{sgn}\big[x(n)\big] - \text{sgn}\big[x(n-1)\big]\Big| \tag{2.3}$$

Where sgn is the signum function. ZCR has shown to have a high correlation to the spectral centroid of a signal, giving an estimate of the dominant frequency. That is, a left-skewed spectrum will yield a lower ZCR than a right-skewed.

Speech is in general composed of altering voiced and unvoiced sounds. In between words small silence periods occur. The voiced sounds in speech are the sounds where a pitch can be found. Unvoiced sounds on the other hand have a structure that resembles noise. Figure 2.2(a) shows 5 seconds of male speech along with the ZCR. The figure shows low ZCR-values where voiced sounds are present and high ZCR-values where unvoiced sounds are present. The altering voiced and unvoiced sounds in speech gives the ZCR-values a relatively large variation. In figure 2.2(b) a zoom of the speech-signal is shown where it is more clear that it is the unvoiced speech parts that gives the large ZCR-values.

In general music is more pitched than speech. This is caused by the clear tones made by the instruments. Figure 2.2(c) shows 5 seconds of music with the corresponding ZCR-values. The figure shows the ZCR does not have as many peaks as the speech-signal. This gives a smaller variation of ZCR. Though, some peaks in the ZCR-plot can be seen. In this case the peak is due to a part in the music where no instruments are playing and only the vocal can be heard. Figure 2.2(d) is a zoom of the music signal. The first part of the plot is where the vocal is dominant and the second part is where there is no vocal and the instruments is the dominant part. The peak in the ZCR-value is due to a unvoiced vocal part.

2.3 Short Time Energy

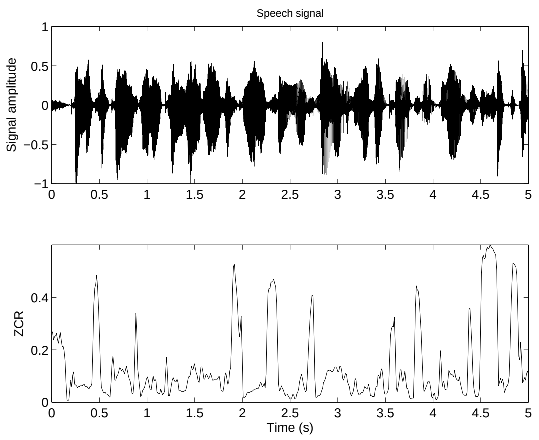
Short time energy (STE) is another simple feature that has been used in various formats in audio applications. STE is defined as the total energy in a frame:

$$\mathrm{STE} = \sum_{n=1}^{N-1} x^2(n) \tag{2.4}$$

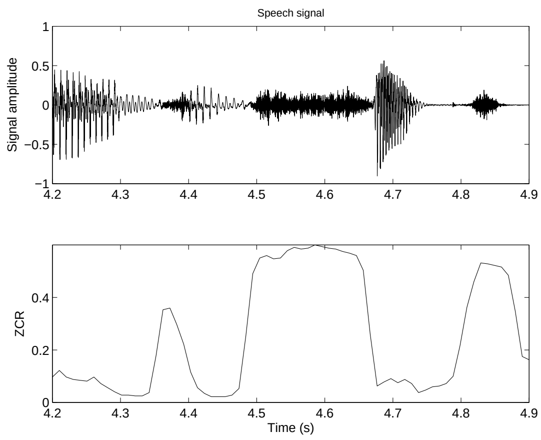
With N as the length of the frame.

As mentioned above speech is composed of altering voiced and unvoiced sounds and silence periods. These unvoiced and silence periods carry less energy than the voiced sounds. Thus, the STE-values for speech will have a large variation. This can also be seen in figure 2.3(a), where the same speech signal as in figure 2.2(a) is shown with the corresponding STE-values. A zoom of this plot is shown in figure 2.3(b), where it can be seen that the voiced speech parts gives larger STE-values than the unvoiced and silence periods.

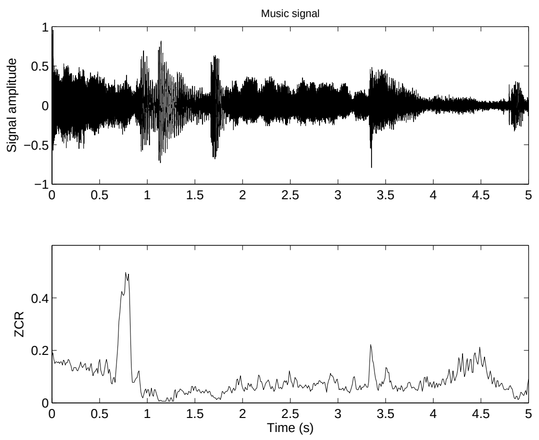
Because of the pitched nature of muzic the STE of music is more constant and larger than speech. This can be seen in figure 2.3(c), where the same music signal as in figure 2.2(c) is shown with the corresponding STE-values. A zoom of figure 2.3(c) is shown in figure 2.3(d), where it is clear that the pitched parts of the music gives high STE-values.



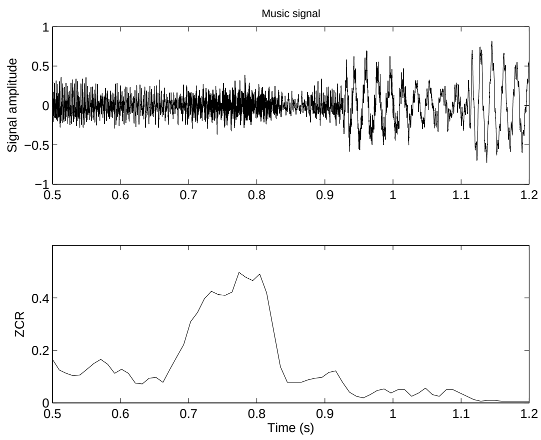
(a) ZCR for 5 seconds of speech.



(b) Zoom of figure 2.2(a).



(c) ZCR for 5 seconds of music.



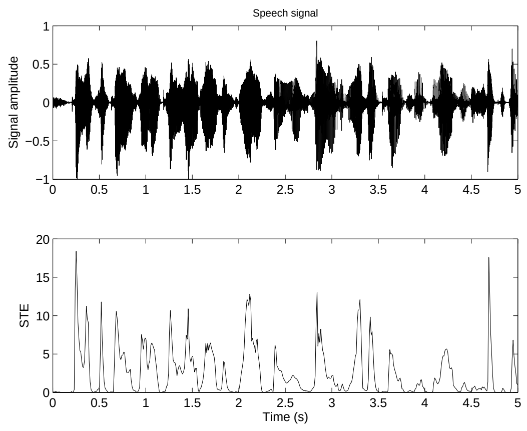
(d) Zoom of figure 2.2(c).

Figure 2.2: (a) shows the ZCR for 5 seconds of speech, where the altering voiced/unvoiced structure is reflected in the ZCR. (b) shows a zoom of (a) where it is more clear that the unvoiced parts of the speech signal gives a high ZCR. In (c) the ZCR for 5 seconds of music is shown, where the ZCR-value is more stable than for speech. In (d) a zoom of the only peak in (c) is shown, and as can be seen, this peak is due to a noise-like sound in the music.

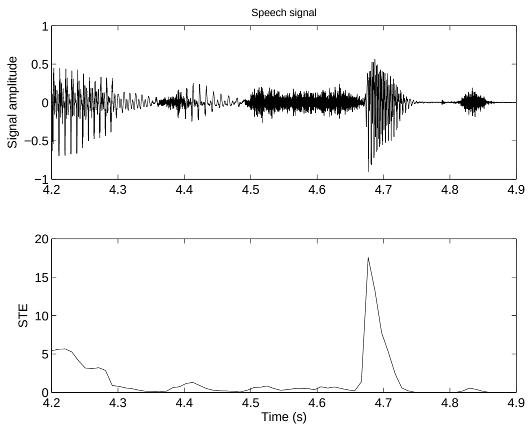
2.4 Spectrum Flux

Spectrum flux (SF) is defined as the change in amplitude spectrum between two successive frames:

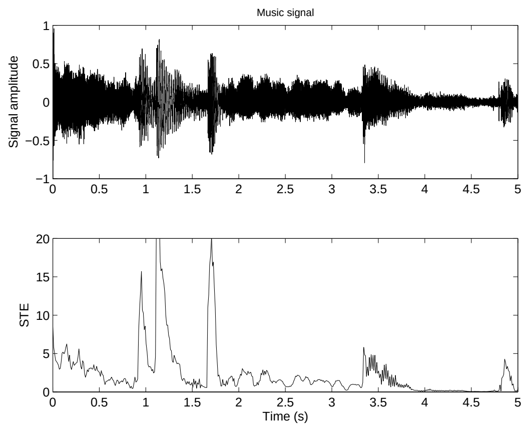
$$\mathrm{SF} = \frac{1}{N} \sum_{k=0}^{N-1} \left[\log\Big(S_i(k)\Big) - \log\Big(S_{i-1}(k)\Big) \right]^2 \qquad (2.5)$$



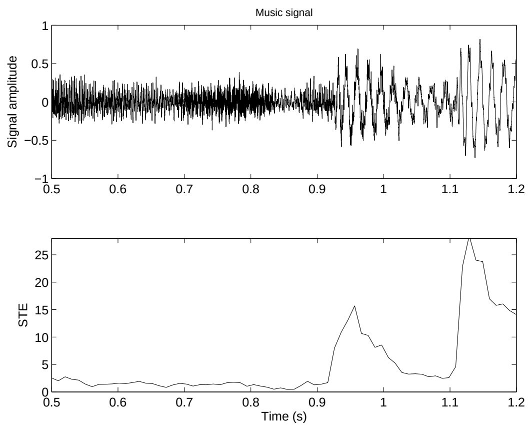
(a) STE for 5 seconds of speech.



(b) Zoom of figure 2.3(a).



(c) STE for 5 seconds of music.



(d) Zoom of figure 2.3(c).

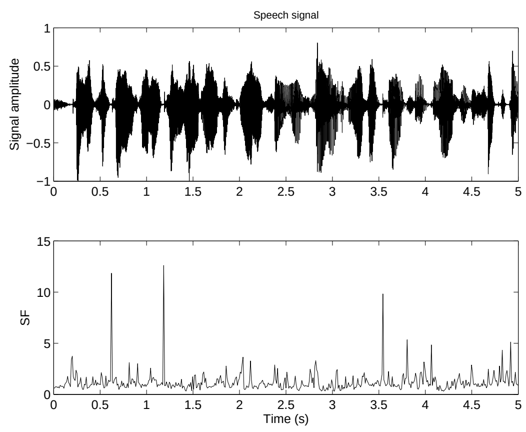
Figure 2.3: (a) shows STE for 5 second of music, where it can be seen that the voiced speech parts gives large STE-values. This is more clear in (b), where a zoom of (a) is shown. (c) shows 5 seconds of music where the STE-values are more constant than for speech. (d) shows a zoom of (c), where it can be seen that the pitched music parts gives large STE-values.

Where $S_i(k)$ is the amplitude spectrum, calculated using the discrete Fourier transform (DFT), of the i'th frame of the input signal:

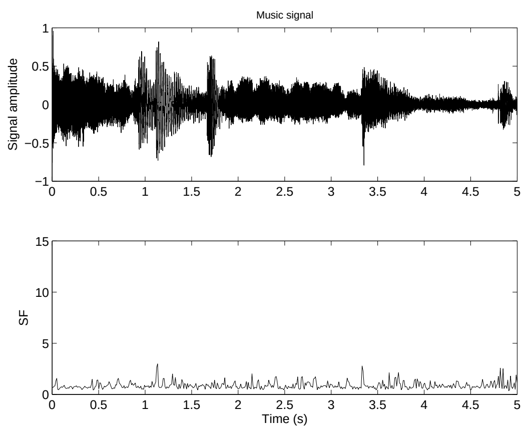
$$S_i(k) = \left| \sum_{n=0}^{N-1} x(n) \exp\left\{ \frac{-2\pi kn}{N} \right\} \right| \tag{2.6}$$

Where N is the length of the frame and $k, n \in [0, N-1]$.

Figure 2.4(a) shows SF for 5 seconds of speech. The altering structure of speech tends to give a large variation of SF. Figure 2.4(b) shows SF for 5 seconds of music. This figure shows that music gives less variation in SF than speech. It is in general the silence parts of the speech that gives large SF-values. Most music does not include such silence parts, thus



(a) SF for 5 seconds of speech.



(b) SF for 5 seconds of music.

Figure 2.4: (a) shows SF for 5 seconds of speech and (b) shows SF for 5 seconds of music. The SF tends to have a larger variation for speech than for music.

SF is more constant for music.

2.5 Cepstrum

Cepstral coefficients were mentioned in section 1.2.1, as features used in audio classification. Initially cepstral analysis was introduced in conjunction with speech recognition, as a way to model the human articulatory system as described below. Later the features have shown useful in speaker recognition as well as other audio applications such as audio classification and music summarization.

As described in [Deller et al., 1993] the speech signal is composed of a quickly varying part $e(n)$(excitation sequence) convolved with a slowly varying part $\theta(n)$ (vocal system impulse response):

$$s(n) = e(n) * \theta(n) \tag{2.7}$$

The convolution makes it difficult to separate the two parts, therefore the cepstrum is introduced. The cepstrum is defined as:

$$c_s(n) = \mathrm{IDFT}\Big\{ \log \big|\mathrm{DFT}\{s(n)\}\big| \Big\} \tag{2.8}$$

Where DFT is the Discrete Fourier Transform and IDFT is the Inverse Discrete Fourier Transform. By moving the signal to the frequency-domain the convolution becomes a multiplication:

$$S(k) = E(k)\Theta(k) \tag{2.9}$$

Further, by taking the logarithm of the spectral magnitude the multiplication becomes an

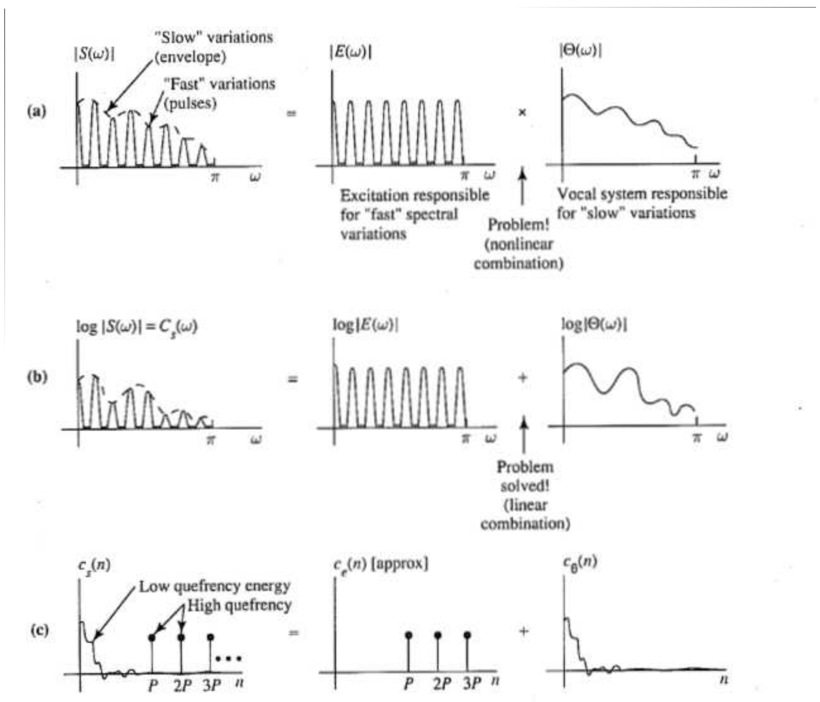

(a)
|S(ω)|
"Slow" variations (envelope)
"Fast" variations (pulses)
|E(ω)|
|Θ(ω)|
Excitation responsible for "fast" spectral variations
Problem! (nonlinear combination)
Vocal system responsible for "slow" variations
(b)
log |S(ω)| = C_s(ω)
log|E(ω)|
log|Θ(ω)|
Problem solved! (linear combination)
(c)
c_s(n)
Low quefrency energy
High quefrency
c_e(n) [approx]
c_θ(n)
P 2P 3P n

Figure 2.5: (a) shows how the signal is composed of a slowly varying envelope part convolved with quickly varying excitation part. By moving to the frequency domain, the convolution becomes a multiplication. Further taking the logarithm the multiplication becomes an addition. Figure taken from [Deller et al., 1993].

addition:

$$\begin{aligned}\log\left|S(k)\right| &= \log\left|E(k)\Theta(k)\right| \\ &= \log\left|E(k)\right| + \log\left|\Theta(k)\right| \\ &= C_e(k) + C_\theta(k) \end{aligned} \tag{2.10}$$

IDFT is linear and therefore works individually on the two components:

$$\begin{aligned} c_s(n) &= \text{IDFT}\{C_e(k) + C_\theta(k)\} \\ &= \text{IDFT}\{C_e(k)\} + \text{IDFT}\{C_\theta(k)\} \\ &= c_e(n) + c_\theta(n) \end{aligned} \tag{2.11}$$

The domain of the signal $c_s(n)$ is called the *quefrency*-domain. Figure 2.5 shows the speech signal transformation process.

2.6 Mel-frequency Cepstral Coefficients

The cepstrum described above has been used with success in speech recognition applications. A further improvement to this method can be obtained by using the mel-weighted cepstrum

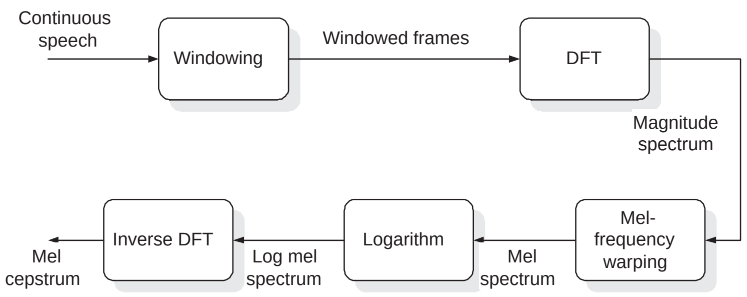


Figure 2.6: MFCC calculation

or mel-cepstrum for short. The mel-cepstrum is calculated in the same way as the real cepstrum except that the frequency scale is warped to the mel scale.

The mel scale is based on an empirical study of the human perceived pitch or frequency. The scale is divided into the mel units. The test persons in the study started out hearing a frequency of 1,000 Hz, which was labeled 1,000 mels for reference. The persons were then asked to change the frequency until they perceived the frequency to be twice the reference. This frequency was then labeled 2,000 mels. This was then repeated with $\frac{1}{10}$, $\frac{1}{2}$, 10 and so on, labeling these frequencies 100 mels, 500 mels, and 10,000 mels. Based on these results a mapping of the linear frequency scale to the mel scale was possible.

The mel scale is generally speaking a linear mapping below 1,000 Hz and logarithmic spaced above. The mapping between the mel frequency scale f_{mel} and the linear scale f is usually done using an approximation, [Feng, 2004]:

$$f_{\text{mel}} = 2595 * \log_{10}\left(1 + \frac{f}{700}\right) \tag{2.12}$$

The calculation of the mel cepstrum leads to the mel-weighted cepstral coefficients (MFCC). The procedure for calculating these coefficients is illustrated in figure 2.6. After the DFT of the signal the magnitude spectrum $S(j)$ is obtained. The mel frequency warping is most conveniently done by utilizing a filter bank with filters centered according to mel frequencies, as seen in figure 2.7. The width of the triangular filters vary according to the mel scale, so that the log total energy in a critical band around the center frequency is included. Let $H_k(j)$ be the sampled representation of the k'th triangular filter. Then the output of the k'th filter is:

$$Y(k) = \sum_{j=1}^{N/2} S(j) H_k(j) \tag{2.13}$$

The last step of the cepstral coefficient calculation is to transform the log of the quefrency domain coefficients to the frequency domain. For this we utilize the IDFT, where N' is the length of the DFT used previously:

$$c(n) = \frac{1}{N'} \sum_{k=0}^{N'-1} \log\Big(Y(k)\Big) \exp\left\{j \frac{2k\pi}{N'} n\right\} \tag{2.14}$$

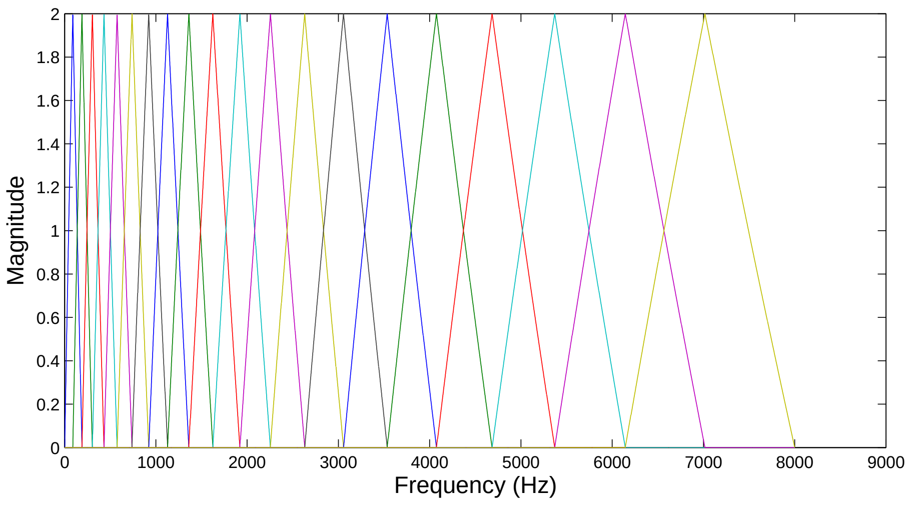


Figure 2.7: Mel spaced filter bank with 20 filters

Which can be simplified, because $Y(k)$ is real and symmetric about $N'/2$, by replacing the exponential by a cosine:

$$c(n) = \frac{1}{N'} \sum_{k=1}^{N'} \log\Big(Y(k)\Big) \cos\left(\frac{2k\pi}{N'} n\right) \tag{2.15}$$

The implementation of MFCCs in our application used the Voicebox toolbox [Brookes, 1998].

Figure 2.8(a) shows 12 MFCCs for 5 seconds of speech and figure 2.8(b) shows 12 MFCCs for 5 seconds of music. The speech shows a large variation in the coefficients. This is due to the altering voiced/unvoiced/silence structure in speech. These different structures has different spectral characteristics, which are reflected in the MFCCs. The MFCCs for the music seems to be much more structured and do not show the same variation in the coefficients.

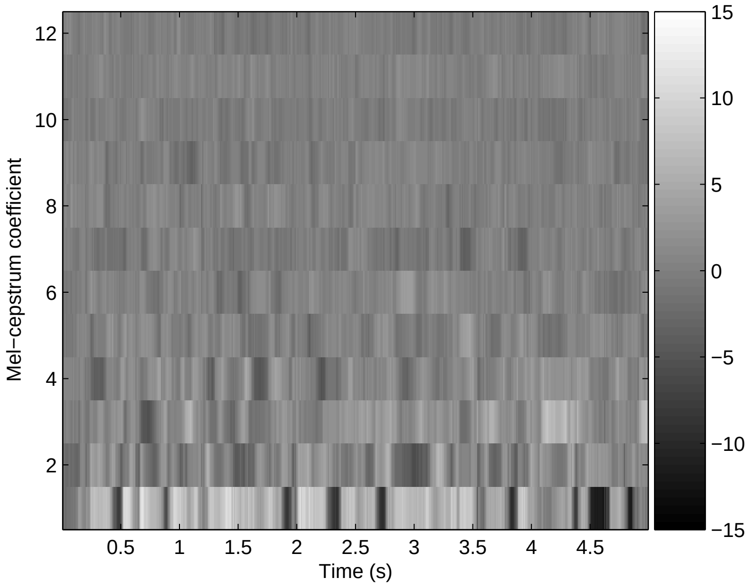
Delta Coefficients

To catch the dynamic changes in the MFCCs the time differentiated or delta coefficients are used. Let $c_i(n)$ be the n'th coefficient extracted at time t_i. Then the delta coefficients are defined as:

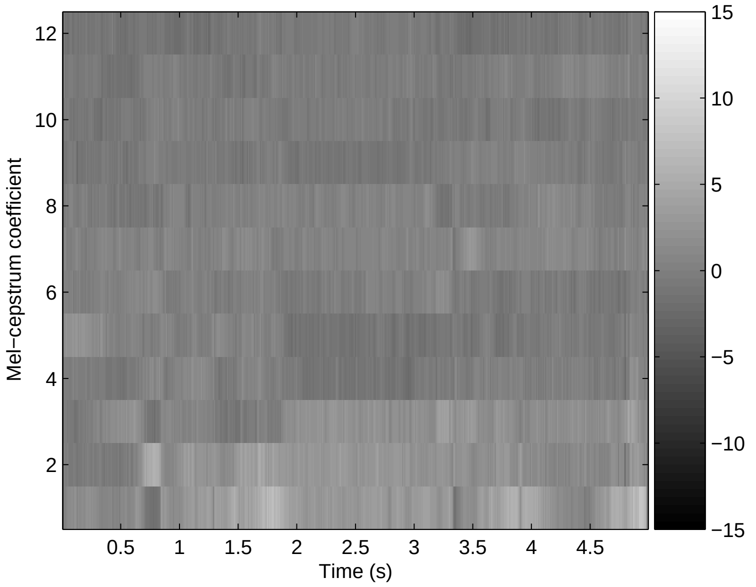
$$\Delta c_i(n) = \frac{1}{2}\Big(c_{i+1}(n) - c_{i-1}(n)\Big) \tag{2.16}$$

Similarly the delta-delta coefficients are defined as:

$$\Delta\Delta c_i(n) = \frac{1}{2}\Big(\Delta c_{i+1}(n) - \Delta c_{i-1}(n)\Big) \tag{2.17}$$



(a) MFCCs for 5 seconds of speech.



(b) MFCCs for 5 seconds of music.

Figure 2.8: (a) shows 12 MFCCs for 5 seconds of speech, and (b) shows 12 MFCCs for 5 seconds of music. The figures shows that the speech has a larger variation in the coefficients than the music does. This is due to the altering voiced/unvoiced/silence structure in speech.

Chapter 3

Audio Classification

The task of segmenting or classifying audio into different audio classes has been implemented using a number of different schemes. The first aspect to consider is the features to use. A multitude of different features have been proposed, based on different observations on the characteristics that separate speech, music and other classes of audio. The simpler features include spectral and time domain features. Some of the most frequently used features are Zero-Crossing Rate (ZCR) and Short Time Energy (STE) [Huang and Hansen, 2004b, Zhang and Jay Kuo, 2001]. Variations of these features were proposed by [Lu et al., 2002], introducing High Zero-Crossing Rate Ration (HZCRR), and Low Short Time Energy Ratio (LSTER). [Scheirer and Slaney, 1997] propose a number of other features including Spectrum Flux (SF), 4 Hz modulation energy, Spectral Centroid, Spectral Rolloff Point.

Cepstral coefficients have been investigated for audio classification for instance in [Li et al., 2001] and [McKinney and Breebaart, 2003], and also utilized in larger automatic broadcast news transcription systems [Gauvain et al., 2002, Hain and Woodland, 1998].

The other aspect to be considered is the classification scheme to use. A number of classification approaches have been proposed, that can be be divided into rule-based and model-based schemes. The rule-based approaches use some simple rules deducted from the properties of the features. [Zhang and Jay Kuo, 2001] use this approach on simple features, yielding a recognition rate of more than 90% for audio classification. As these methods depend on thresholds, they are not very robust to changing conditions, but may be feasible for real-time implementations.

Model-based approaches such as GMM classifiers are employed frequently for instance in [Gauvain et al., 2002, Hain et al., 1998, Scheirer and Slaney, 1997]. [Scheirer and Slaney, 1997] combine a number of features for 1 second windows and achieve a recognition rate in simple speech/music classification of about 94%. The classification is done using Gaussian Maximum A Posteriori classifier, GMM, K-NN, and a spatial partitioning classifier based on k-d trees. The results show that these classifiers perform almost equally well. The correct classification rate rose to 98.6% for a 2.4 seconds window. Introducing a third class containing speech mixed with music reduced the classification rate to 65%.

[McKinney and Breebaart, 2003] investigate four feature sets, including modulation energies of different frequency bands. The feature sets are: (1) Low-level signal parameters including RMS-level, spectral centroid, bandwidth, ZCR, spectral roll-off frequency, pitch (2)MFCCs giving a feature set containing 13 DC values of the MFCCs and the modulating energies as above. (3) psychoacoustic features including roughness, loudness and sharpness; and (4) an auditory model representation of temporal envelope fluctuations. For classification quadratic discriminate analysis was used on 7.5 seconds frames. The classification result for general audio (5 classes including speech, music, and noise) is $92 \pm 3\%$.

Some studies has been performed using clean samples, e.g. using windows only containing speech *or* music, [Li et al., 2001] investigate classification on mixed data. They investigate 143 mean and variance features, based on temporal and spectral features, MFCCs, LPCs. The classification is done on continuous audio data, obtained by recording tv-news shows. MFCC-based features are reported to perform best yielding an overall classification rate of $\simeq$ 95% using 6 classes (speech, music, noise, and 3 classes of noisy speech) and a simple Gaussian MAP classifier.

The systems mentioned above observe features over windows of up to 7.5 s, calculating variances to catch the varying nature of the audio. Hidden Markov Models (HMM) on the other hand try to model the time sequence of the features to take this into account [Kemp et al., 2000], sometimes combined with GMMs as output probabilities. HMMs are employed in [Vandecatseye and Martens, 2003] to classify audio into 5 acoustic classes. The features used are MFCCs and pitch. The classification of speech yields a correct classification rate of 99.5%. The switching between different audio classes is also considered in [Huang and Hansen, 2004b] with a network of GMMs based on heuristics achieves an accuracy of 96%.

In this project we have chosen to limit the number of audio classes to consider speech and music. This is based on the observation that our speech recognition system [1] can handle both noise and silence segments. In addition music is often used as separators for *logically* coherent speech segments. Therefore the main purpose of this audio classification is to extract as much speech as possible.

This chapter is organized as follows: First we review the theory behind different classification models used in this chapter. Next we introduce the audio-database used for the audio classification. Then we show how the features are adapted for the audio classification. We then compare the different classification models followed by feature selection. Finally we examine the performance of the selected model and features.

3.1 Classifiers

A supervised classifier works by modelling the posterior probability $p(C_k|\mathbf{x})$ that an input vector $\mathbf{x}$ belongs to class C_k. *Generative* classifiers assume that the input density function

[1]presented in chapter 5

$p(\mathbf{x})$ has a specific functional form, which depends on a set of parameters $\boldsymbol{\theta}$. Optimizing the parameters $\boldsymbol{\theta}$ to a given training set $\mathcal{D}$ can be done using either maximum likelihood or Bayesian inference. Once the class-conditional probability $p(\mathbf{x}|C_k)$ and the prior probability $P(C_k)$ have been estimated, Bayes theorem can be used to calculate the posterior probability:

$$p(C_k|\mathbf{x}) = \frac{P(C_k)p(\mathbf{x}|C_k)}{p(\mathbf{x})} \tag{3.1}$$

Discriminative classifiers do not assume a specific functional form for the input density function, but model the posterior probability $p(C_k|\mathbf{x})$ directly.

In general it is not possible to say in advance which kind of classifier is the best. The performance of the classifiers depend on the amount of training data, as well as the nature of the data. Another aspect to be considered is the computational aspects, i.e. the runtime of the training algorithm and of the classification.

In the following section we will describe different discriminative and generative classifiers, which are later compared in the work of classifying audio into speech and music.

For reference we define a training set $\mathcal{D} = \{(\mathbf{x}^n, \mathbf{t}^n)\}$ of corresponding input-vectors $\mathbf{x}^n$ and class labels $\mathbf{t}^n$, where $\mathbf{x} = (x_1, ..., x_d)^\top$, and $\mathbf{t} = (t_1, ..., t_c)^\top$ is a 1-of-c decoded vector with $t_k = 1$ if $\mathbf{x}$ belongs to class C_k and zero otherwise. d is the dimension of the input space and c is the number of output classes.

3.1.1 Discriminative Classifiers

Linear Discriminant

The linear discriminant defines an output function $y_k(\mathbf{x})$ of the input-vector $\mathbf{x}$ for each output class $\mathcal{C}_k$. With y_k given by:

$$\begin{aligned} y_k(\mathbf{x}) &= w_{k0} + \sum_{i=1}^{d} w_{ki} x_i \\ &= w_{k0} + \mathbf{w}_k^\top \mathbf{x} \end{aligned} \tag{3.2}$$

Where w_{k0} is the bias and $\mathbf{w}_k$ is the weight vector. A given input sample $\mathbf{x}$ is then assigned to class C_k if $y_k(\mathbf{x}) > y_j(\mathbf{x})$, $j \neq k$. By including the bias in the weight vector and setting $\mathbf{x} = (1, x_1, ..., x_d)$ the notation becomes:

$$\begin{aligned} y_k(\mathbf{x}) &= \sum_{i=0}^{d} w_{ki} x_i \\ &= \mathbf{w}_k^\top \mathbf{x} \end{aligned} \tag{3.3}$$

The weights $\mathbf{w}_k$ are found by minimizing the sum-of-squares error function for the given

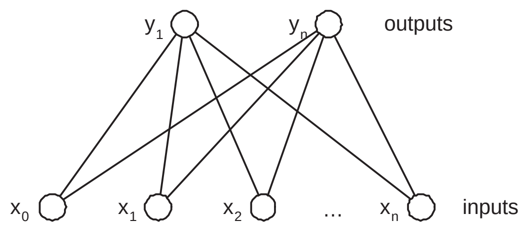


Figure 3.1: Graphical interpretation of linear discriminant. The linear discriminant can be seen as a 1-layer neural network with linear activation functions. x_0 constitutes the bias

training set $\mathcal{D} = \{(\mathbf{x}^n, \mathbf{t}^n)\}$ with N samples:

$$E(\mathbf{w}_k) = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{c}\left\{\mathbf{w}_k^\top \mathbf{x}^n - t_k^n\right\}^2 \tag{3.4}$$

Defining $\mathbf{X}$ with dimension $N \times c$ and elements x_i^n, $\mathbf{W}$ with dimension $c \times d$ and elements w_{ki} and $\mathbf{T}$ with dimension $N \times c$ and elements t_k^n equation 3.4 can be rewritten:

$$E(\mathbf{W}) = \frac{1}{2}(\mathbf{W}^\top \mathbf{X}^\top \mathbf{X}\mathbf{W} + \mathbf{T}^\top \mathbf{T} - 2\mathbf{W}^\top \mathbf{X}^\top \mathbf{T}) \tag{3.5}$$

By setting the derivative of equation 3.5 w.r.t. $\mathbf{W}$ to zero gives the normal equations for the least-squares problem:

$$(\mathbf{X}^\top \mathbf{X})\mathbf{W} = \mathbf{X}^\top \mathbf{T} \tag{3.6}$$

Solving for $\mathbf{W}$ gives:

$$\mathbf{W} = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{T} = \mathbf{X}^\dagger \mathbf{T} \tag{3.7}$$

Where $\mathbf{X}^\dagger$ is the pseudo inverse of $\mathbf{X}$.

The linear classifier trained using the sum-of-squares error function is very simple to implement and can be trained very quickly, because of the closed form solution. In general the sum-of-squares error function is not optimal for classification problems because it was derived from maximum likelihood problems where targets are Gaussian distributed variables. The classification with 1-of-c decoded outputs on the other hand yields binary values, which differs clearly from the Gaussian approximation.

Neural Network Classifier

The linear discriminant presented above could be seen as a one-layer linear network. We have also investigated an artificial neural network (designated NN below). The network used in this project is a two-layer fully connected feed-forward network. The model contains n_I input units, n_H hidden units, and n_c output units. The graphical representation is seen in figure 3.2.

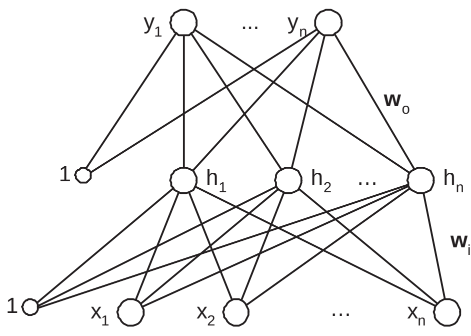


Figure 3.2: Graphical interpretation of a neural network

The functions evaluated in each unit are given below. The hidden layer utilizes tanh as activation function i.e. $\psi = \tanh(\cdot)$, while the output layer uses a linear activation function.

$$h_j(\mathbf{x}) = \psi\left(\sum_{l=1}^{n_I} w_{jl}^I x_l + w_{j0}^I\right) \tag{3.8}$$

$$\phi_i(\mathbf{x}) = \sum_{j=1}^{n_H} w_{ij}^O h_j + w_{i0}^O \tag{3.9}$$

The output of output unit k of the neural network should model the posterior probability of the k'th class $p(C_k|\mathbf{x})$, To interpret the outputs of the neural network as probabilities they should consequently lie in the range [0,1] and sum to 1. This is accomplished by applying the SoftMax function on the outputs:

$$y_i = \frac{\exp(\phi_i)}{\sum_{j=1}^{c}\exp(\phi_j)}, i = 1, 2, ..., c \tag{3.10}$$

Training of the NN to model the classification problem does not have a closed form solution like the linear discriminant. Instead the training algorithm adjusts weights using gradient descent methods utilizing the backpropagation algorithm described in [Bishop, 1995, chap. 4].

Using gradient descent for NN training basically starts with an initial set of weights $\mathbf{w}^{(0)}$ chosen randomly. The weights are changed iteratively, i.e. the weights at the j'th step $\mathbf{w}^{(j)}$ is used for: $\mathbf{w}^{(j)} = \mathbf{w}^{(j-1)} + \eta\Delta\mathbf{w}^{(j)}$. Using the gradient descent approach the step $\Delta\mathbf{w}^{(j)}$ is chosen as the negative gradient with respect to the error function.

$$\Delta\mathbf{w}^{(j-1)} = -\nabla E\mid_{\mathbf{w}^{(j-1)}} \tag{3.11}$$

Obtaining the gradient for the weights in the network is done through the backpropagation algorithm. The basis of the backpropagation procedure is to look at the derivative of the error E^n with respect to a weight w_{ji}:

$$\frac{\partial E^n}{\partial w_{ji}} = \frac{\partial E^n}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}} \tag{3.12}$$

Using the chain rule for partial derivative, as we know that the error only depends on the weights w_{ji} through the summed input to the unit j. Introducing the notation $\delta_j \equiv \frac{\partial E^n}{\partial a_j}$ and calculating the derivative $\frac{\partial a_j}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_i w_{ji} z_i = z_i$, we can write (3.12) as

$$\frac{\partial E^n}{\partial w_{ji}} = \delta_j z_i \tag{3.13}$$

For the output units this gives:

$$\delta_k \equiv \frac{\partial E^n}{\partial a_k} = \psi'(a_k) \frac{\partial E^n}{\partial y_k} \tag{3.14}$$

And for the hidden units:

$$\delta_j \equiv \frac{\partial E^n}{\partial a_j} = \sum_k \frac{\partial E^n}{\partial a_k} \frac{\partial a_k}{\partial a_j} \tag{3.15}$$

This leads to the backpropagation formula:

$$\delta_j = \psi'(a_j) \sum_k w_{kj} \delta_k \tag{3.16}$$

In general the complexity and convergence of the backpropagation algorithm means that training a neural network is quite time consuming. The performance of the optimization can be improved by employing second order algorithms (Levenberg-Marquardt, Gauss-Newton, pseudo-Gauss-Newton). These algorithms utilize a quadratic approximation of the error surface using a Taylor expansion of the second order term of the cost function E around $\hat{w}$:

$$E(\mathbf{w}) = E(\hat{\mathbf{w}}) + (\mathbf{w} - \hat{\mathbf{w}})' \mathbf{g}_{\hat{\mathbf{w}}} + \frac{1}{2} (\mathbf{w} - \hat{\mathbf{w}}) \mathbf{H}_{\hat{\mathbf{w}}} (\mathbf{w} - \hat{\mathbf{w}}) + \mathcal{O}(\mathbf{w}^3) \tag{3.17}$$

$g_{\hat{\mathbf{w}}}$ is the first order gradient of the cost function in $\hat{\mathbf{w}}$ and $H_{\hat{\mathbf{w}}}$ is the second order derivative - the Hessian - of the cost function in $\hat{\mathbf{w}}$. The first order derivative of $\mathbf{w}$ is:

$$\nabla E(\mathbf{w}) = \mathbf{g}_{\hat{\mathbf{w}}} + \mathbf{H}_{\hat{\mathbf{w}}} (\mathbf{w} - \hat{\mathbf{w}}) \tag{3.18}$$

We want to find a local minimum $\mathbf{w} = \mathbf{w}_0$, i.e. finding: $\nabla E(\mathbf{w}_0) = 0$, which means that we now can isolate $\mathbf{w}_0$:

$$\mathbf{w}_0 = \hat{\mathbf{w}} + (-\mathbf{H}_{\hat{\mathbf{w}}}^{-1} \mathbf{g}_{\hat{\mathbf{w}}}) \tag{3.19}$$

Taking this step is the (full) Newton algorithm. Evaluating the full Hessian and calculating the inverse $\mathbf{H}_{\hat{\mathbf{w}}}^{-1}$ is a complex operation. Therefore easier methods use an approximation to the Hessian, for instance the pseudo-Gauss-Newton only uses the diagonal elements of the Hessian (here for each variable w_i in $\mathbf{w}$):

$$\Delta w_i = -\frac{\partial E}{\partial w_i} \Big/ \frac{\partial^2 E}{\partial w_i^2} \tag{3.20}$$

As mentioned earlier the sum-of-squares error function are linked to the assumption that targets are normal distributed. However, as $\mathbf{y}$ models a probability and $\mathbf{t}$ is a 1-of-c decoded target vector, the Gaussian error is inappropriate. We can write the conditional likelihood function:

$$p(\mathbf{t}^n|\mathbf{x}^n) = \prod_k^c (y_k^n)^{t_k^n} \tag{3.21}$$

This will give the more appropriate *cross-entropy* cost function [see Bishop, 1995, p. 237]. To optimize we need the negative log-likelihood which, for an entire data set of independent samples, is:

$$E_{\text{entropy}} = -\sum_n^N \sum_k^c t_k^n \ln y_k^n \tag{3.22}$$

For a two-class problem with just a single output this reduces to:

$$E_{\text{entropy}} = -\sum_n^N \Big(t_k^n \ln y_k^n + (1 - t_k^n) \ln(1 - y_k^n) \Big) \tag{3.23}$$

Using the entropy error function yields some simple derivatives for the use in the backpropagation algorithm.

Differentiating the cross-entropy error function yields quite a simple derivative:

$$\frac{\partial E_{\text{entropy}}}{\partial y^n} = \frac{y^n - t^n}{y^n(1 - y^n)} \tag{3.24}$$

Which makes the back-propagation of entropy-based errors feasible.

Neural Network Pruning

A property of the neural network is that it tends to overfit to the training data, which is of course a problem of generalization. Because of this tendency a method has been developed to make neural networks more generalizable. The method is called pruning, and is based on the observation that some of the weights in a neural network are less significant than others. A popular pruning method is called *Optimal brain damage*, which utilizes the concept of *saliency* of a weight.

Saliency is a measure of the importance of the weight. The saliency is estimated by changing a weight w_i to $w_i + \delta w_i$. This gives a change in the error function E:

$$\delta E = \sum_i \frac{\delta E}{\delta w_i} \delta w_i + \frac{1}{2} \sum_i \sum_j H_{ij} \delta w_i \delta w_j + \mathcal{O}(\delta w^3) \tag{3.25}$$

Using the Hessian matrix defined above:

$$H_{ij} = \frac{\partial^2 E}{\partial w_i \partial w_j} \tag{3.26}$$

Using the diagonal approximation of the Hessian and observing that the first term of (3.25) should vanish if training has converged and discarding the higher order term $\mathcal{O}(\delta w^3)$ gives:

$$\delta E = \frac{1}{2} \sum_i \sum_j H_{ij} \delta w_i \delta w_j = \frac{1}{2} \sum_i H_{ii} {\delta w_i}^2 \tag{3.27}$$

If a weight is removed, i.e. set to zero, the change in the error function can be approximated by setting $\delta w_i = w_i$ in (3.27). The saliency for a weight w_i is thereby approximated to $H_{ii}{w_i}^2/2$. This measure can be used to remove the least salient weights. The pruning can be done by the following procedure:

1. Train a reasonably large network using some appropriate stopping criterion
2. Compute the second order derivatives H_{ii} for each weight w_i, to calculate the saliency $H_{ii}{w_i}^2/2$
3. Remove a number of the weights with lowest saliencies
4. Retrain the network and repeat from step 2.

This approach may both improve the speed of the trained network *and* generalizability performance.

K-Nearest Neighbor Classifier

The K-Nearest Neighbor (K-NN) classifier is an example of a non-parametric classifier. The basic algorithm in such classifiers is simple. For each input-vector $\mathbf{x}$ to be classified, the K nearest training samples are found, and then $\mathbf{x}$ is assigned to the class C_k if the number of training samples included in K from this class is larger than for all other classes: $K_k > K_j$, for all $j \neq k$. Euclidean distance is usually used to calculate the distance between two input-vectors $\mathbf{x} = \{x_1, ..., x_d\}$ and $\mathbf{y} = \{y_1, ..., y_d\}$:

$$d_E(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2} \tag{3.28}$$

For the special case of $K = 1$ we will obtain the nearest neighbor classifier, which simply assigns the input-vector to the same class as that of the nearest training vector.

The K-NN algorithm, is very simple yet rather powerful, and used in many applications. The use of the Euclidian distance makes the assumption that each dimension is equally distributed. If one input dimension has larger values than the other dimensions, this dimension will dominate in the sum of distances. The solution to this problem could be to normalize the feature sets.

To sum up the K-NN classifier has some qualities that are important such as

- It requires no training and this is helpful especially when new training data is added.
- Uses local information and hence can learn complex functions without the need to represent them explicitly, which is demonstrated in figure 3.3

While the disadvantages are

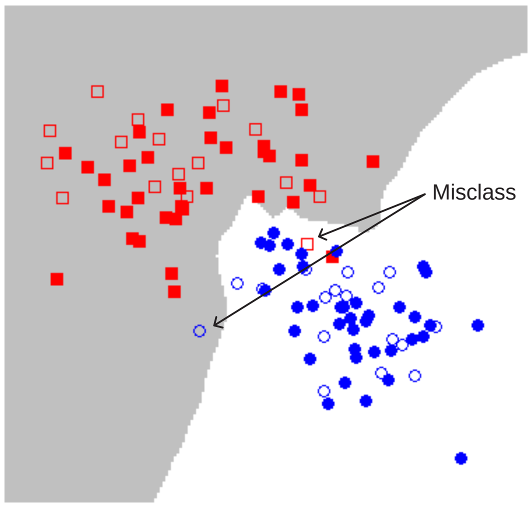


Figure 3.3: Illustration of the knn-classifier using 3 nearest neighbors. Filled samples represent the training set that are used to construct the class boundaries shown by the grey area.

- The classifier needs to store the entire training set for use when a new input-vector is to be classified.
- The classification time is longer when compared to other classifiers.

The storage problem and slow speed of the algorithm can be counteracted by performing vector quantization on the data, to reduce the number of training samples.

3.1.2 Generative Classifiers

The last classifier considered is the Gaussian Mixture Model, which is the only generative classifier considered.

Gaussian Mixture Model

The basis of the Gaussian Mixture Model is to combine a number of Gaussian distributions using a linear combination of the components. So the probability density function is given as:

$$p(\mathbf{x}) = \sum_{j=1}^{M} p(\mathbf{x}|j)P(j) \tag{3.29}$$

Let each component density be a multivariate Gaussian distribution, given by:

$$p(\mathbf{x}|j) = \frac{1}{(2\pi)^{n/2}|\mathbf{\Sigma}_j|^{1/2}} \exp\left\{ -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_j)^\top \mathbf{\Sigma}_j^{-1}(\mathbf{x}-\boldsymbol{\mu}_j)\right\} \tag{3.30}$$

If we constrain the covariance matrices to be scaled unit matrices: $\mathbf{\Sigma} = \sigma_j^2\mathbf{I}$ the densities becomes:

$$p(\mathbf{x}|j) = \frac{1}{(2\pi\sigma_j^2)^{d/2}} \exp\Big\{ - \frac{(\mathbf{x}-\boldsymbol{\mu}_j)^2}{2\sigma_j^2}\Big\} \tag{3.31}$$

$P(j)$ is the prior probability that the data-point $\mathbf{x}$ is generated from component j.

The maximum likelihood solution for the parameters $\boldsymbol{\mu}_j$, σ_j^2, and $P(j)$ given a training set $\mathcal{D}$ can be determined using the expectation-maximization (EM) algorithm, described in more detail in [Bishop, 1995, Chap. 2]. This gives the following update equations:

$$\boldsymbol{\mu}_j^{\text{new}} = \frac{\sum_{n=1}^N P^{\text{old}}(j|\mathbf{x}^n)\mathbf{x}^n}{\sum_{n=1}^N P^{\text{old}}(j|\mathbf{x}^n)} \tag{3.32}$$

$$(\sigma_j^{\text{new}})^2 = \frac{1}{d}\frac{\sum_{n=1}^N P^{\text{old}}(j|\mathbf{x}^n)(\mathbf{x}^n - \boldsymbol{\mu}_j^{\text{new}})^2}{\sum_{n=1}^N P^{\text{old}}(j|\mathbf{x}^n)} \tag{3.33}$$

$$P(j)^{\text{new}} = \frac{1}{N}\sum_{n=1}^{N} P^{\text{old}}(j|\mathbf{x}^n) \tag{3.34}$$

The parameters are updated iteratively, where the 'old' parameters are used to calculate the 'new' parameters.

3.2 Audio Database

For the audio classification a database of speech and music was collected. The speech contains a wide range of different radio shows. The samples are chosen to reflect many different kinds of typical speech, ranging from anchor[2] speakers in almost perfect conditions to conversations between multiple speakers. Also, narrow-band telephone interviews are present. Some of the speech clips contain speech from reporters speaking from noisy environments. 50 speech clips are gathered and the length of each clip is 45 seconds. The sources of the clips are listed in table 3.1. The clips were collected in October and November 2005.

A collection of music clips has also been made. The choice of music has been done to reflect a diverse range of genres. This is to make our classifiers versatile for the broad range of music that could occur on a radio show. 50 music clips each 45 seconds long were collected. The clips cover the 10 genres listed in table 3.1.

3.3 Feature Integration

The features presented in chapter 2 make the basis for the speech/music classification.

[2]Hosts of news shows are often called news anchors, we will therefore use this term for host speakers.

Speech	
no. of pieces	Description
21	CNN Hourly News Update
1	60 minutes
5	BBC radio 4 talk radio
3	CBS What's in news
3	CNN Biz update
3	NPR 'talk radio'
4	Sportsweek
3	Talking 6 Music

Music	
no. of pieces	Description
5	rock
5	easy listening
5	country
5	jazz
5	electronica
5	alternative
5	reggae
5	latin
5	rb&soul
5	pop&dance

Table 3.1: Speech and music clips in the audio-database

The time resolution of the features was set to 20 ms. Classification based solely on such short intervals is not very feasible. Instead schemes for integrating the features for longer windows have shown more useful. A simple method for such an integration is to calculate means and variances for a larger window. More elaborate methods such autocorrelation of features have also been utilized [Meng et al., 2005]. The use of mean and variance have shown useful in [Saunders, 1996] and [Li et al., 2001], so we have chosen to use this approach.

The size of the window (which we will call the decision-window) to integrate over depends on a number of factors. The optimal size of the decision-window has been discussed in the literature. Saunders [1996] for instance showed that a 2.4 seconds decision-window was feasible.

In general the length should be long enough to yield adequate precision of the statistics, and maybe catch some of the longer term characteristics. On the other hand the length should be short enough to yield an adequate resolution. In the work by [Scheirer and Slaney, 1997] on speech/music classification it is indicated that a 1 second window is feasible. We have thus chosen to work with 1 second decision-window.

In chapter 2 a feature set of 29 features was presented. The feature set included: ZCR, STE, SF, 13 MFCC, and 13 ΔMFCC. The 0'th MFCC is identical to the log energy. Using these features the means and variances are calculated over the decision-window giving 58 integrated features.

In addition to the means and variances more elaborate features have been developed, for instance [Lu et al., 2002] proposed the high zero-crossing rate ratio (HZCRR). The definition is:

$$\text{HZCRR} = \frac{1}{2N}\sum_{n=1}^{N}\Big[\text{sgn}(\text{ZCR}(n) - \frac{3}{2}\text{MZCR}) + 1\Big] \tag{3.35}$$

I.e. the HZCRR is the ratio of the number of feature-windows whose ZCR-value is above 1.5 times the average ZCR in a decision-window. The motivation for this feature is that the ZCR has more peaks in speech compared to music.

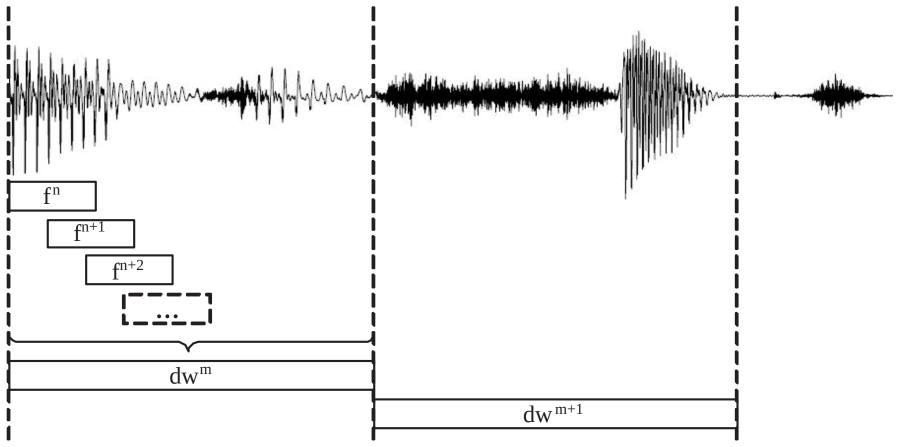


Figure 3.4: The windowing involved in classification feature extraction. The feature frames are used to calculate the basic features. Features from the frames are then summed up in a decision window (dw) giving the features used for classification.

A similar reasoning was done for the STE-feature. The definition of the low short-time energy ratio (LSTER), which is built on the fact that speech generally contains less energy than music. It is defined as the ratio of the number of feature-windows whose STE are less than 0.5 times of average short time energy in a decision-window:

$$\text{LSTER} = \frac{1}{2N} \sum_{n=1}^{N} \left[\text{sgn}\Big(\frac{1}{2}\text{MSTE} - \text{STE}(n)\Big) + 1 \right] \tag{3.36}$$

In all the feature integration leads to 60 features for each decision-window. The features are listed in table 3.2 and the hierarchy is schematically shown in figure 3.5.

3.3.1 Feature Space Exploration

In this section we will explore the feature space graphically. The exploration of the 60-dimensional feature space is using principal component analysis (PCA). PCA is in short a linear transform that can be used to transform data onto a smaller number of dimensions, retaining as much as possible of the variance of the data. PCA is for instance presented in [Bishop, 1995, chap. 8]. Projecting features onto the largest principal components, we are able to observe the features in the directions of most variability.

First we consider the time and spectral features. Figure 3.6(a) shows a PCA-plot using the 8 derived features from these three basis features. This plot does show some separability but a large overlap of distribution is seen. Figure 3.6(b) shows the PCA-plot of the features derived from the MFCCs. This figure shows a clearer separation between the two classes. Figure 3.7 shows the PCA-plot using all 60 features. This figure shows that the separability is better for the combination of all 60 features.

Though, the PCA-plot does not show the correlation between the different features. Thus, it

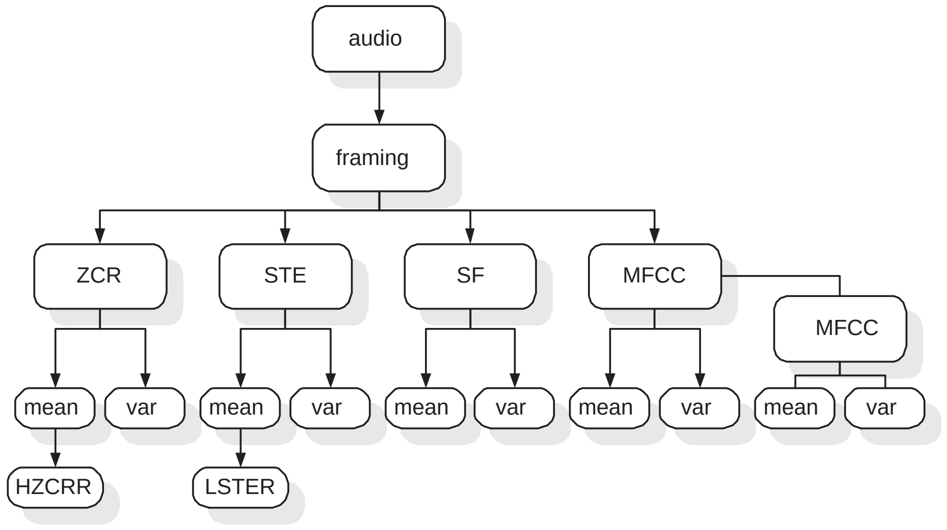


Figure 3.5: Schematic presentation of feature calculation showing the dependencies among the 60 features.

is not easily seen which features are redundant. Even though MFCC-based features exhibited better separability than the other features, it is clear that the addition of the basic features improved the separability.

Generally it is not easily seen which features and how many are most feasible for the task, as the number of combinations is insurmountable. Graphically inspecting all combinations of the 60 features is simply not manageable. Consequently methods for qualified feature selection are needed and introduced in later sections.

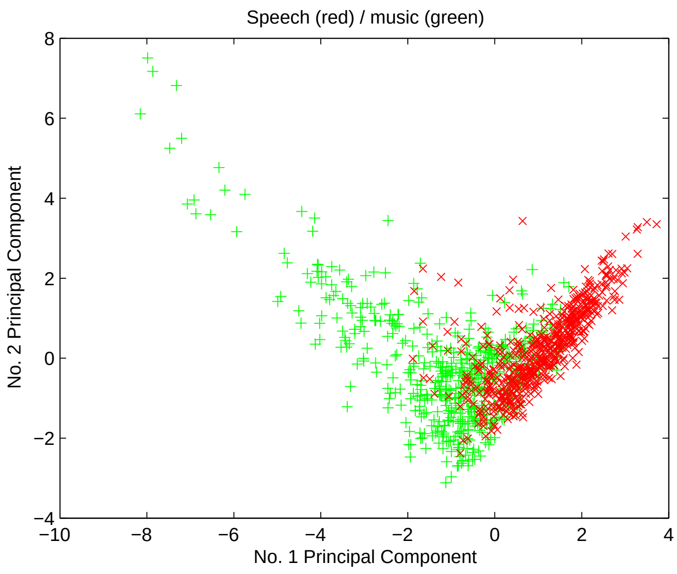
The feature extraction includes one last step, namely normalization. The features were normalized to have zero-mean and unit variance.

3.4 Classifier Evaluation

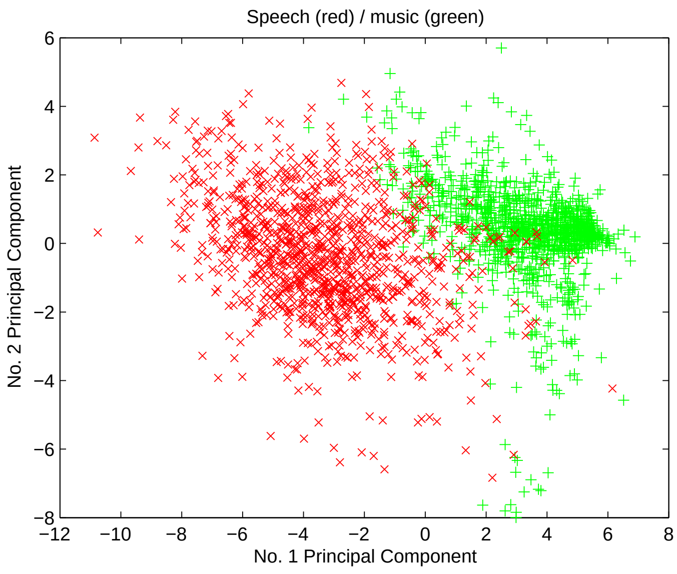
Section 3.1 described four classification models. In this section they are applied to the speech/music classification problem, to determine their applicability for the problem.

3.4.1 Test Setup

The evaluation of the classifiers was done using a 10-fold cross-validation setup. The audio-database was divided into 10 evenly sized subsets, each subset containing 225 seconds of speech and 225 seconds of music. The training was then performed on 9 subsets and validated on the remaining subset. This was then repeated by leaving each of the ten subsets out and training on the remaining 9 subsets.



(a) ZCR, STE, and SF-based features.



(b) MFCC-based features.

Figure 3.6: PCA-plots of the features for the two subsets, namely the 8 time- and spectral-based features (a) and the 52 MFCC and ΔMFCC-based features (b) described in section 3.3. Red samples represent music and green represent speech.

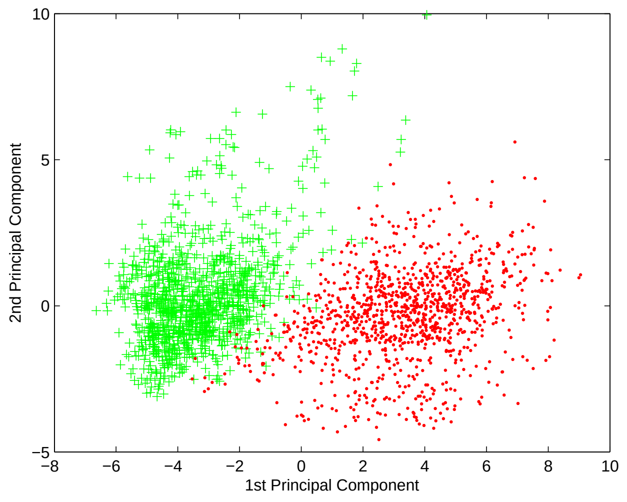


Figure 3.7: PCA-plot of all 60 features. Red samples represent music and green represent speech. The addition of the ZCR, STE, and SF-features seems to increase the separation that was already present using only MFCCs

Feature no.	Description	Short
1	Variance of zero-crossing rate	VZCR
2	Variance of spectrum flux	VSF
3	Variance of short-time energy	VSTE
4	Mean of zero-crossing rate	MZCR
5	Mean of spectrum flux	MSF
6	Mean of short-time energy	MSTE
7	High zero-crossing rate ratio	HZCRR
8	Low short-time energy ratio	LSTER
9	Mean of log energy	
10-21	Mean of MFCC 1-12	
22	Mean of delta log energy	
23-34	Mean of delta MFCC 1-12	
35	Variance of log energy	
36-47	Variance of MFCC 1-12	
48	Variance of delta log energy	
49-60	Variance of delta MFCC 1-12	

Table 3.2: Features used in the classification experiments. The numbers are sometimes used for easier reference in some of the experiments/figures but textual or abbreviated forms will mainly be used.

A feature set consisting of the 60 features was extracted from the audio on a 1 second basis as described in section 3.3. This means that each subset contained 450 labelled patterns.

In the tests the classifiers were trained as follows:

- **Linear discriminant**: The weights of the classifier was found by minimizing the sum-of-squares error function.
- **NN** : The weights was found by performing 5 gradient descent iterations followed by 50 pseudo-Gauss-Newton iterations. The NNwas implemented using the DTU:Toolbox [Kolenda et al., 2002]
- **GMM** : Training the GMMs was done by performing 150 EM-iterations. The covariances were constrained to be diagonal matrices.
- **K-NN** : No training needed, except saving the training data.

GMMs and K-NN was implemented using the Netlab toolbox [Nabney and Bishop, 2004].

The evaluation of the classifiers was done by varying the complexity of the models.

3.4.2 Results

Linear Discriminant

A mean training misclassification rate of 2.1% with 0.2% standard deviation, and a mean test misclassification of 2.6% with 2.1% standard deviation was obtained using the linear discriminant.

Neural Network

Figure 3.8 shows training and test misclassification rate for the NN-classifier as function of number of hidden units. The test misclassification rate is relatively constant with a mean value of approximately 3%. No significant improvement was obtained by increasing the number of hidden units. The figure shows a very low training misclassification rate, indicating that the NN-classifier tends to overfit the training data.

Gaussian Mixture Model

Figure 3.9 shows training and test misclassification rate for the GMM-classifier as function of number of mixture components for each of the two classes. As expected the mean misclassification rate on the training sets decreased when the number of mixture models was increased. The best misclassification rate on the test sets is observed using 11 mixture models for each class. Increasing the number of mixture models beyond 11 the model overfits the training data and the test misclassification rate tends to increase.

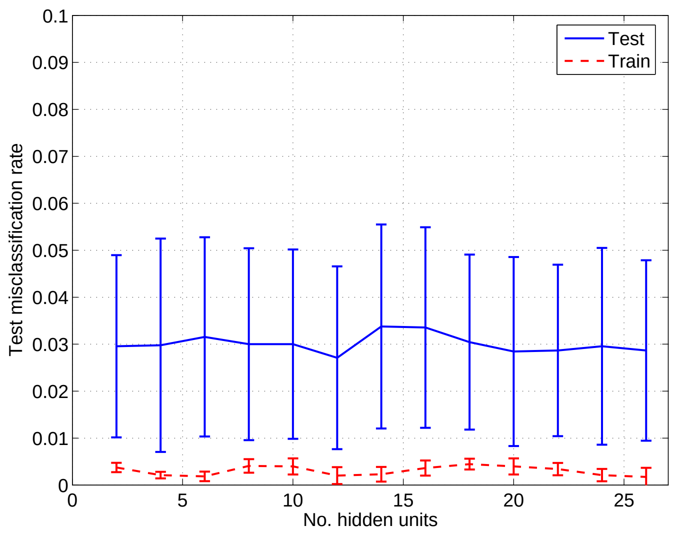


Figure 3.8: Training and test misclassification rate for the NN-classifier as function of number of hidden units used in the NN. The figure shows mean and standard deviation for 10 runs using the cross-validation setup.

KNN

Figure 3.10 shows the test misclassification rate as function of the number of neighbors (K) to consider in the K-NN-classifier. Setting $K = 5$ or $K = 7$ minimizes the mean test misclassification rate. This gives a mean test misclassification rate of 3.4% with standard deviation of 2.5%.

Classifier Comparison

Table 3.3 summarizes the best results obtained with the four classifiers.

All four classifiers showed very good results. A bit surprisingly the linear discriminant showed the best test performance. On the other hand a very low mean misclassification rate on the training set for the NN-classifier indicates that the training data has been overfitted. The linear discriminant does not, because of the simplicity of the decision boundaries, exhibit the same tendency to overfit the training data. This causes a relatively high training misclassification rate but a lower test misclassification. The GMM-classifier does not show the same stability as the other classifiers and the performance depends more clearly on the choice of number of mixtures used.

Another important issue to consider when choosing a classifier is how the misclassifications are distributed. In our system it is important that as much speech as possible is classified correct. This is primarily because wrongly classified speech is impossible to recover, while music is not crucial in the speech recognition. On the other hand the segmentation of the audio stream should be as precise as possible as processing music in the speech recognition

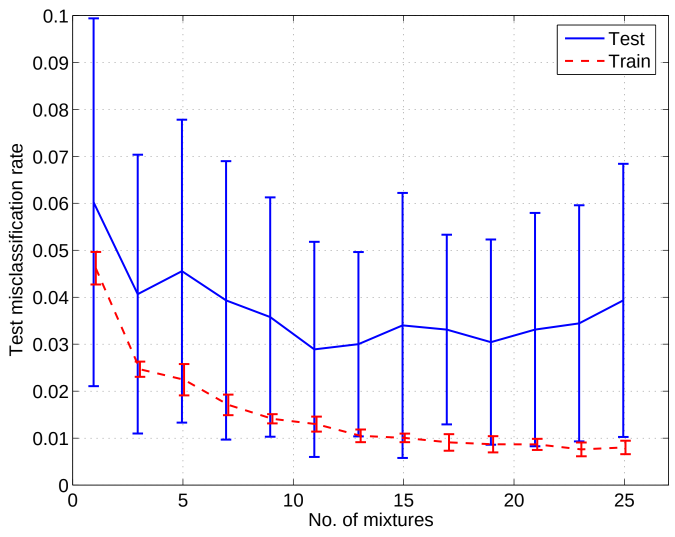


Figure 3.9: Training and test misclassification rate for the GMM-classifier as function of number mixture models used for each of the two classes. The figure shows mean and standard deviation for 10 runs using the cross-validation setup.

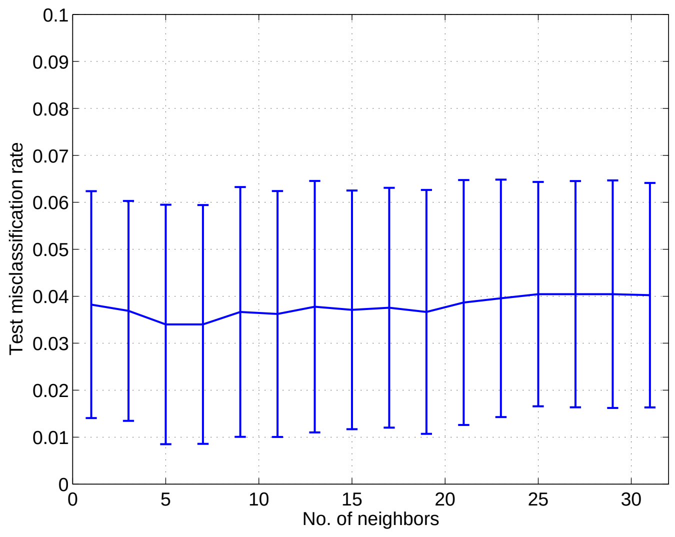


Figure 3.10: Training and test misclassification rate for the K-NN-classifier as function of number of neighbors to be considered in the K-NN-classifier. The figure shows mean and standard deviation for 10 runs using the cross-validation setup.

	mean test	std test	mean train	std train
Linear	0.026	0.021	0.020	0.002
NN	0.027	0.020	0.002	0.002
GMM	0.029	0.017	0.015	0.002
K-NN	0.034	0.025		

Table 3.3: Results obtained with the four classifiers: Linear discriminant, NN, GMM, and K-NN. The table shows mean and standard deviation using the cross-evaluation setup for the test and training sets. The performance of the four classifiers are comparable, though K-NN gives a higher misclassification rate than the three other classifiers.

Linear	pred speech	pred music
true speech	2149	101
true music	16	2234

NN	pred speech	pred music
true speech	2193	57
true music	68	2182

GMM	pred speech	pred music
true speech	2204	46
true music	104	2146

K-NN	pred speech	pred music
true speech	2137	113
true music	40	2210

Table 3.4: Confusion matrices for the four classification models.

system is very time demanding and should be avoided.

Table 3.4 shows the confusion matrices for a test run for the four classifiers. Most of the misclassification of the linear discriminant is due to predicting speech as music. The NN-classifier has a more even distribution of misclassification among the two classes. The GMM classifies most speech correct among the four classifiers but classifies a high number of music samples as speech. Similar to the linear discriminant, most of the misclassifications of the K-NN-classifier are due to predicting speech as music.

Based on these observation we have chosen to focus on the NN and the linear discriminant in the following sections.

3.5 Feature Evaluation

Section 3.4 showed that very good classification results can be obtained using the feature set consisting of the 60 proposed features. The very high dimensionality of the feature space may require very large training sets, cf. 'the curse of dimensionality'. It is therefore natural to find smaller feature sets easing the model training and remove redundant features.

This section introduce two approaches to reduce the number of features. The first approach is based on pruning a NN. The second approach uses a sequential search of the features using the linear discriminant.

3.5.1 Neural Network Pruning Scheme

Initially pruning of the NN was inferred to investigate if the method could be made more robust, i.e. reduce the overfitting problem. These initial runs showed that some of the features were consistently pruned out. Therefore we developed a procedure to do an initial pruning of the features so that the obviously redundant features could be removed. This procedure is step 1 of our NN-based feature evaluation.

Step 2 takes a more exhaustive approach and tries to find the best combination of the features that were left after the observations in step 1.

The following subsections describe the procedures used in two steps.

Step 1

The algorithm used in this step is shown in **Algorithm 1** and is called PRUNINGSTEP1. Basically the procedure trains a fully connected NN and prunes this network. Hereby the network with the lowest test error is found. The features that are not pruned out in this network are stored.

Each iteration of the PRUNINGSTEP1 starts out with a call to TRAIN&PRUNENET.

Each run of TRAIN&PRUNENET consist of the following: A fully connected NN is trained with 5 gradient descent iterations followed by 50 pseudo-Gauss-Newton iterations. Then the 2 lowest saliency weights are pruned out and the network is retrained. This procedure is repeated until all weights are pruned out. After each training period the network is tested on a test set, and the set of features $\widehat{\mathbf{F}}_i$, used in the network giving the lowest test misclassification rate are returned. Likewise the number of hidden units Nh_i used in the best network is stored.

The PRUNINGSTEP1 algorithm is setup using the feature set $\mathbf{F}$, which contains all 60 features. When a feature is not present in $\widehat{\mathbf{F}}_i$ in three succeeding iterations, the feature is deleted from the feature set, and thereby not considered in the remaining iterations.

To reduce the complexity of the network topology, we have inferred a small heuristic to reduce the number of hidden units between iterations. In the first three iterations 14 hidden units are used. In the remaining iterations the average number of hidden units in the results of the last three iterations.

The features in the feature set after 100 iterations of the PRUNINGSTEP1 algorithm is used in the set of features used by the pruning algorithm in step 2.

The pseudo-code for the PRUNINGSTEP1 algorithm is showed in **Algorithm 1** and the pseudo-code for the TRAIN&PRUNENET is showed in **Algorithm 2**.

Algorithm 1 pruningStep1

```
$\mathcal{D}_{\text{train}} \leftarrow \{(\mathbf{x}_{\text{n}}^{\text{train}}, \mathbf{t}_{\text{n}}^{\text{train}})\}$
$\mathcal{D}_{\text{test}} \leftarrow \{(\mathbf{x}_{\text{n}}^{\text{test}}, \mathbf{t}_{\text{n}}^{\text{test}})\}$
$\mathbf{F} \leftarrow f_1, f_2, .., f_n$
$\text{Nh} \leftarrow 14$

for i=1:100 do
    $\widehat{\mathbf{F}}_i, \text{Nh}_i \leftarrow$ train&pruneNet$(\text{Nh}, \mathbf{F}, \mathcal{D}_{\text{train}}, \mathcal{D}_{\text{test}})$
    for each $f_n \in \mathbf{F}$ do
        if $f_n \notin (\widehat{\mathbf{F}}_{i-2} \cup \widehat{\mathbf{F}}_{i-1} \cup \widehat{\mathbf{F}}_i)$ then
            $\mathbf{F} \leftarrow \mathbf{F} \setminus \{f_n\}$
        end if
    end for
    $\text{Nh} \leftarrow \lfloor < \text{Nh}_n > \rfloor, n \in [i-2, i]$
end for
return $\mathbf{F}$
```

Algorithm 2 train&pruneNet$(\text{Nh}, \mathbf{F}, \mathcal{D}_{\text{train}}, \mathcal{D}_{\text{test}})$

```
$\mathbf{N} \leftarrow$ fully connected neural network
$\text{minErr} \leftarrow \infty$

while numberOfWeights$(\mathbf{N}) > 2$ do
    $\mathbf{N} \leftarrow$ gradientDescent$(\mathbf{N}, \mathcal{D}_{\text{train}})$
    $\mathbf{N} \leftarrow$ pseudoGaussNewton$(\mathbf{N}, \mathcal{D}_{\text{train}})$
    if errRate$(\mathbf{N}, \mathcal{D}_{\text{test}}) < \text{minErr}$ then
        $\text{minErr} \leftarrow$ errRate$(\mathbf{N}, \mathcal{D}_{\text{test}})$
        $\mathbf{N}_{\text{best}} \leftarrow \mathbf{N}$
    end if
    $\mathbf{N} \leftarrow$ prune2weights$(\mathbf{N})$
end while

$\mathbf{F}_{\text{out}} \leftarrow$featuresUsed$(\mathbf{N}_{\text{best}})$
$\text{Nh}_{\text{out}} \leftarrow$numberOfHiddenUnitsUsed$(\mathbf{N}_{\text{best}})$
return $\mathbf{F}_{\text{out}}, \text{Nh}_{\text{out}}, \text{minErr}$
```

Algorithm 3 PRUNINGSTEP2

```
$\mathcal{D}_{\text{train}} \leftarrow \{(\mathbf{x}_n^{\text{train}}, \mathbf{t}_n^{\text{train}})\}$
$\mathcal{D}_{\text{test}} \leftarrow \{(\mathbf{x}_n^{\text{test}}, \mathbf{t}_n^{\text{test}})\}$
$\mathbf{F} \leftarrow$ PRUNINGSTEP1
$\text{Nh} \leftarrow 4$
$\text{maxErr} \leftarrow 0$

while SIZEOF($\mathbf{F} > 1$) do
    for each $f_n \in \mathbf{F}$ do
        $\widehat{\mathbf{F}} \leftarrow \mathbf{F} \setminus \{f_n\}$
        $E_{\text{rate}} \leftarrow$ TRAIN&PRUNENET($\text{Nh}, \widehat{\mathbf{F}}, \mathcal{D}_{\text{train}}, \mathcal{D}_{\text{test}}$)
        if $E_{\text{rate}} > \text{maxErr}$ then
            $f_{\text{del}} \leftarrow f_n$
        end if
    end for
    $\mathbf{F} \leftarrow \mathbf{F} \setminus \{f_{\text{del}}\}$
end while
```

Step 2

Step 1 should indicate that a number of the features could be left out without any loss of classification performance. The purpose of step 2 is to make a more exhaustive search of the remaining features to find the optimal feature set.

The initial feature set of step 2 contains the features selected from step 1. Basically this step is done uses backward elimination. That is, the network is trained leaving out one feature from the feature set. The altering training and weight pruning procedure TRAIN&PRUNENET is done in the same way as is step 1. The feature which is let out in the model giving the lowest test misclassification rate is deleted form the feature set. This procedure is repeated until two features are left. The pseudo-code for the PRUNINGSTEP2 algorithm is shown in **Algorithm 3**.

3.5.2 Neural Network Pruning Results

In this section the results from running the NN pruning algorithms are presented.

The audio-database was divided into two evenly sized subsets SET1 and SET2. The pruning procedures are very time demanding, thus the cross-evaluation used for the classifier evaluation has not been utilized for this setup.

Step 1 Results

The PRUNINGSTEP1 algorithm was run 10 times, 5 times with SET1 used as training set and SET2 as test set, and 5 times vice versa.

Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10
Nh=4	Nh=4	Nh=5	Nh=2	Nh=6	Nh=4	Nh=2	Nh=2	Nh=3	Nh=3
3.20%	2.93%	2.80%	3.07%	2.80%	2.80%	1.91%	2.22%	2.22%	2.00%
1	1	1		1					
2	2	2	2	2	2	2	2	2	2
	3								
4	4	4		4			4		
					5	5	5	5	5
6	6	6		6		6			
9	9	9	9	9	9		9		
10	10	10	10	10	10	10	10	10	10
11	11	11	11	11	11		11		11
12	12	12	12	12	12				12
					14				
					20				
35	35	35	35	35	35	35	35	35	35
36	36	36	36	36	36	36	36	36	36
38	38	38	38	38			38		38
	41								
		48				48	48	48	48
49			49		49	49		49	49
	51				51	51	51	51	51
52	52	52	52	52	52	52	52	52	52
	55	55	55	55		55	55		
57	57	57		57					

Table 3.5: This table shows the results from 10 runs of the pruningStep1 algorithm. The first 5 runs are using set1 as training set and set2 as test set, and vice versa for the last 5 runs. The table shows the number of hidden units used in the final model along with the test misclassification rate. Also the features left after 100 iterations are shown. The features numbers are defined in table 3.2

Table 3.5 shows the features left in the models after 100 iterations of the pruningStep1 algorithm. The table also shows the misclassification rate and the number of hidden units used in the network giving the lowest misclassification rate.

The evolution of each run is shown in appendix B.

The average misclassification rate in the final models is 2.96% when using set1 as training and 2.23% when using set2. This should be compared with the misclassification rate when using all 60 features as input to the model, where set1 gives a misclassification rate of 3.36% and set2 gives a misclassification rate of 2.15%. Thus, the reduction in the feature set gives a minor decrease in the misclassification rate when using set1 and a minor increase when using set2.

The final models after 100 iterations use between 2-6 hidden units. The number of features used in the final model are decreased from 60 to 9-17 features.

When looking at the features left it can be seen that a large number of features only appear once or never appears. These features are listed in table 3.6. In general the features based on the higher order MFCC are deleted in the pruning process. Only the variances of ΔMFCC-4, ΔMFCC-7, and ΔMFCC-9 are left among the MFCC-features with order 4-12.

Feature no.	Description
3	var STE
7	high ZCR ratio
8	low STE ratio
13-21	mean MFCC4-12
22	mean delta log energy
23-34	mean ΔMFCC1-12
37	var MFCC2
39-47	var MFCC4-12
50	var ΔMFCC2
53-54	var ΔMFCC5-6
56	var ΔMFCC8
58-60	var ΔMFCC10-12

Table 3.6: Features that are pruned out 9 or 10 times in 10 runs of pruningStep1

Feature no.	Description
2	var SF
10	mean MFCC1
35	var log energy
36	var MFCC1
52	var ΔMFCC4

Table 3.7: Features never deleted in 10 runs of pruningStep1

Some features are never deleted in the 10 runs of the pruningStep1 algorithm. These features are listed in table 3.7.

Step 2 Results

As earlier mentioned the purpose of step 2 is to make a more exhaustive search of the remaining features. The goal is to find the best combination of the features.

All features that survived one or more runs of pruningStep1 are used in the initial set of features used in pruningStep2. This yields in total 22 features.

The pruningStep2 has been run 20 times, 10 times with each data-set as training. Figure 3.11 shows the mean test misclassification rate along with the standard deviation for the 10 runs for each training set as function of the number of features used in the model. The figures show that the number of features can be reduced to 10 without any loss in test misclassification.

The figure also shows a clear difference between the two sets. When testing on set2 the misclassification is larger which suggests that the set contains some kind of outlier.

Table B.1 in appendix B shows the order the features was deleted from the feature set for each of the 20 runs of the pruningStep2 algorithm. This table shows that there is no

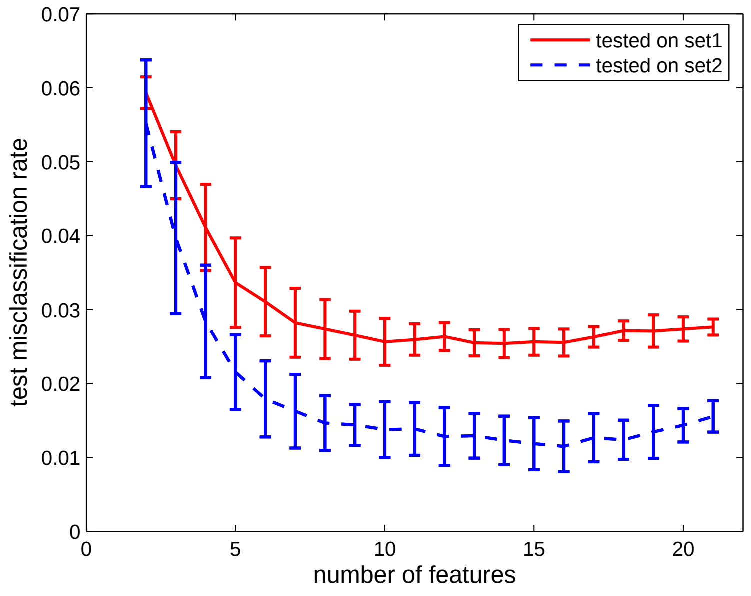


Figure 3.11: The figure shows the mean test misclassification rate and standard deviation for 10 runs of the PRUNINGSTEP2 algorithm for each of the two training sets.

consistency in the choice of features. Very similar results can be obtained using different combinations of the features.

3.5.3 Feature Evaluation Using Linear Discriminant

The linear discriminant is trained a lot faster than the NN. Therefore another approach is taken when doing the feature evaluation using the linear discriminant. Two approaches has been considered, namely the backward elimination and forward selection schemes [Bishop, 1995].

In the backward elimination scheme all 60 features are considered in the initial setup. Each feature is in turn left out of the feature set, and the linear discriminant is trained and evaluated on the test set. The feature that is left out of the model giving the lowest misclassification rate on the test set is deleted from the feature set. This procedure is repeated until the feature set is empty.

The forward selection scheme starts out with an empty feature set. In turn each feature is temporarily added to the feature set and the linear discriminant is trained and evaluated on the test set. The feature added to the model giving the best test misclassification rate is then permanently added to the feature set. The procedure is repeated until all features have been added to the feature set.

The 10-fold cross-validation scheme described in section 3.4.1 is also used in this setup. This means that the test misclassification rate is taken as a mean over 10 different test sets.

Results

Figure 3.12(a) shows the misclassification rate for the training and test sets using the backward selection scheme. Figure 3.12(b) shows the results for the forward selection scheme. The figures show marginal difference, though placing the test curves in the same plot shows that the backward selection gives marginally better performance. This can be seen in figure 3.13. The figure shows that using the linear discriminant a performance of 3% misclassifications is attainable using only 8 features.

Table 3.8 shows the ranking order of the features using the two different approaches. The first line shows the order in which the features were selected using the forward selection scheme. The second line shows the reverse order in which the features were pruned in the backward selection scheme. The 14 best features are shown, and only 6 features are in common in the two approaches.

3.5.4 Test Discussion

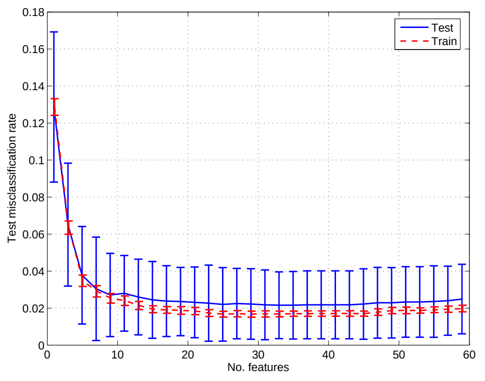
The purpose of this section has been to reduce the number of features used in the model without reducing the performance of the model. Two approaches have been considered. The NN-approach showed that the number of features could be reduced from 60 to about 10 and still obtain a test misclassification rate below 3%. Section 3.5.2 showed that several feature combinations can obtain this result

Using the linear discriminant two other feature reduction schemes were investigated, namely the backward elimination and the forward selection scheme. Also these two schemes showed ambiguity in the feature selection. Despite this ambiguity the results obtained using the different schemes were very similar. This again shows that several feature combinations can obtain almost equal results.

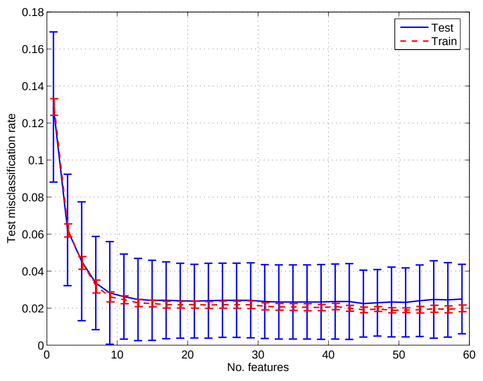
3.6 Final Model Evaluation

The past sections have described and analyzed different classifiers. Also the features have been investigated. PCA-plots showed that speech and music is well separated in feature space. It was shown that the number of features used could be reduced from 60 to about 10-14 features without any loss in test misclassification rate. The linear discriminant showed very good performance and no improvement was obtained using the more complex NN-classifier.

Because of its simplicity, the linear discriminant has been chosen for the speech/music classification part of our system. In this section the properties of the linear discriminant are investigated.



(a) Backward elimination



(b) Forward selection

Figure 3.12: Misclassification rate on test sets as function of number of features for the backward elimination scheme(a) and the forward selection scheme(b).

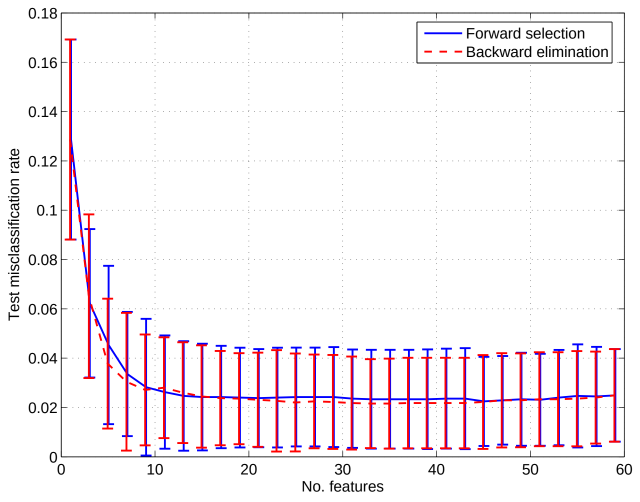


Figure 3.13: Misclassification rate on test sets as function of number of features for the backward elimination and forward selection scheme. The figure shows marginally difference between the two approaches.

Forward selection	36	38	35	52	54	10	2	9	44	12	15	58	45	42
Backward elimination	36	9	10	38	52	5	57	8	12	3	6	15	11	19

Table 3.8: The first line shows the order in which the features were selected in the forward selection scheme. The second line shows the reverse order the features were deleted from the backward elimination scheme. Only the 14 best features are listed.

3.6.1 Features

Section 3.5.3 showed that the number of features could be reduced without any loss in test misclassification rate. It was also shown that several combinations of the features gave similar results. For the final classifier the 14 best features according to the backward elimination scheme for the linear discriminant are chosen. The features are listed in table 3.9 with the first feature being the feature with the highest rank, that is the feature that was deleted last in the backward elimination scheme.

In figure 3.14 the histograms for the 6 highest ranking features are shown. The largest separation between the two classes can be found in feature 36, 38, and 52. Though, these feature probability densities are much similar. The other features show less separation.

In figure 3.15 a PCA-plot using the final 14 features are shown. This figure shows that the two classes are well separated in feature space.

3.6.2 Training Set Size

An important property of a classifier is the amount of training data needed to make a fully reliable model. This section investigates the amount of training data needed for the linear discriminant.

The audio-database was divided into two sets, each set containing 1125 seconds of speech and 1125 seconds of music. The training set was enlarged by choosing random data from one of the sets. The model was tested using the other set.

Figure 3.16 shows the test misclassification rate as function of the amount of training data used. This figure shows that the test misclassification has stabilized using 500 seconds of training data.

3.6.3 Training and Run Time

A nice feature of the linear classifier is its fastness. Because of the closed form solution to the weight settings, no iterative training procedure is needed. Thus, once the features has been extracted it takes less than 1 second to train and test a model with 4500 seconds of audio.

3.6.4 Error Investigation

The confusion matrix for the linear discriminant using all 60 features was shown in table 3.4. This table showed that most of the misclassifications was due to classifying speech as music. Table 3.10 shows the confusion matrix for the linear discriminant using the selected 14 features. In this table the audio-database has been divided into two evenly sized sets used

Feature no.	Description
36	var MFCC-1
9	mean log energy
10	mean MFCC-1
38	var MFCC-3
52	var ΔMFCC-4
5	mean SF
57	var ΔMFCC-9
8	low STE ratio
12	mean MFCC-3
3	var STE
6	mean STE
15	mean MFCC-6
11	mean MFCC-2
19	mean MFCC-10

Table 3.9: The features chosen for the final classification model. The features are chosen according to the backward elimination scheme for the linear discriminant. The first feature is the feature with the highest rank.

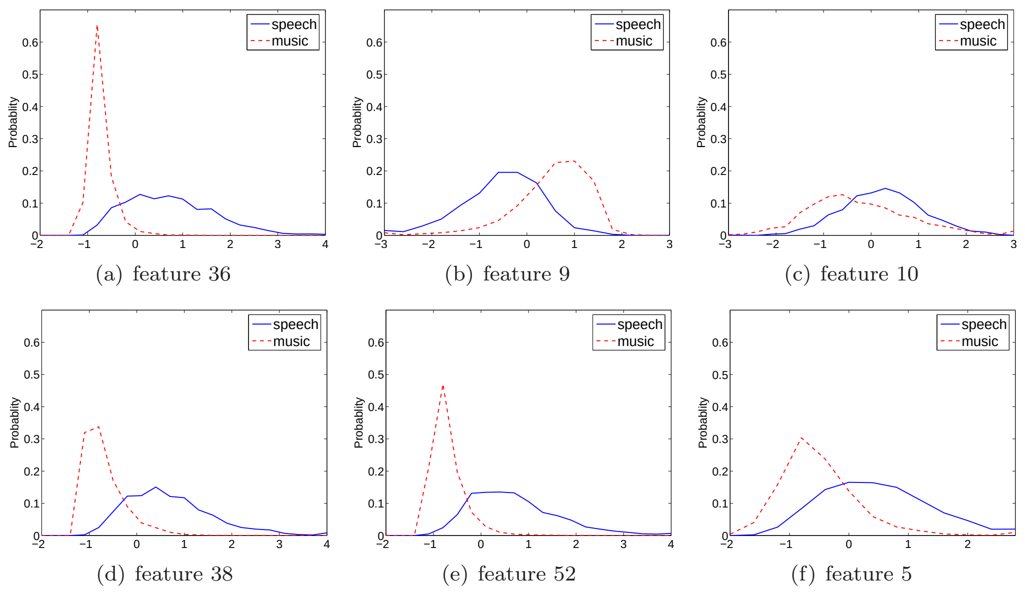


Figure 3.14: Histograms for the 6 features with highest rank according to the backward elimination scheme for the linear discriminant.

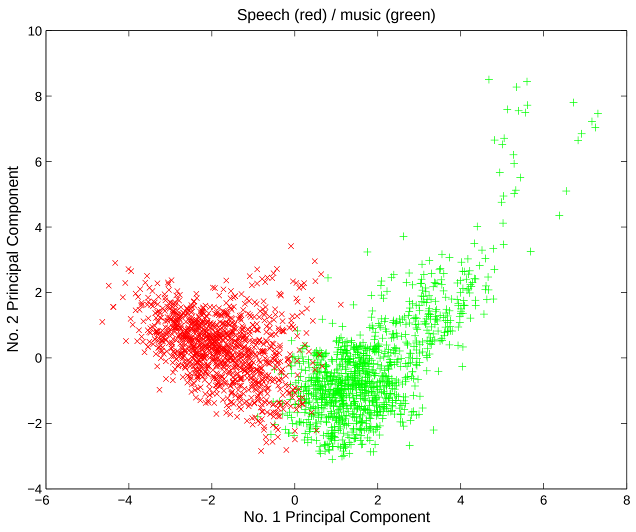


Figure 3.15: PCA-plot using the 14 features used in the final classifier.

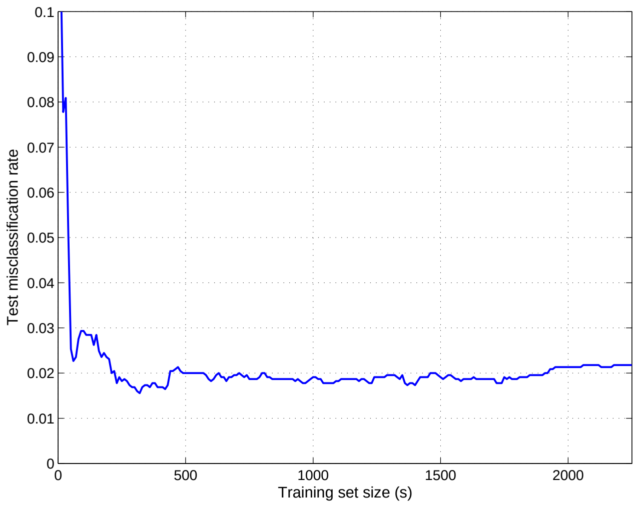


Figure 3.16: This figure shows the test misclassification rate as function of the size of the training set.

linear	pred speech	pred music
true speech	1082	43
true music	6	1119

Table 3.10: Confusion matrix for the linear discriminant.

linear	pred speech	pred music
true speech	1111	14
true music	23	1102

Table 3.11: Confusion matrix using the biased linear discriminant.

for training and test respectively. An overall test misclassification rate of 2.2% is observed. Again, most of the misclassifications are due to classifying speech as music.

As earlier described, it is more crucial to classify speech correct than classifying music correct. If the softmax function is applied at the outputs of the linear discriminant the outputs can be seen as probabilities. Then, by applying the rule that the audio is classified as speech if the probability of speech exceeds 0.45. This biased classifier gives the confusion matrix shown in table 3.11. This table shows that the misclassification rate for speech can be reduced without increasing the overall misclassification rate. In fact, the overall misclassification rate was reduced to 1.6% in this case.

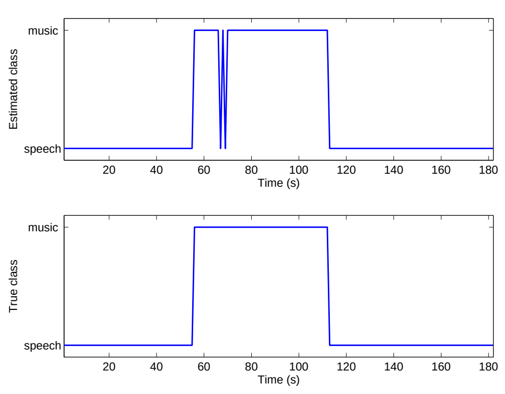
An investigation of the misclassifications showed that when speech was classified as music it was mainly due to two kinds of error sources. The main error source is speakers are in a noisy environment, for instance with reporters reporting from the field. This background noise makes it difficult for the system to catch the characteristics of speech. The other error source is laughter. Some speech parts in the audio-database contains small segments of laughter, and this was classified as music by the classifier. However it is not crucial to classify laughter as non-speech, as it indeed is non-speech and no word information is lost when removing laughter.

The music parts classified as speech mainly consist of parts where no instruments are playing. Some of these parts are vocal parts in the music. These parts in general have the same altering voiced/unvoiced/silence structure as speech, which makes it difficult for the system to classify correct.

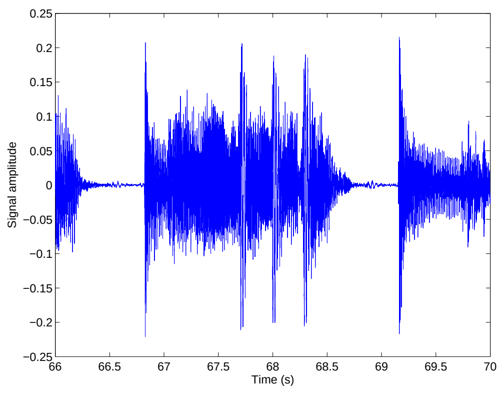
3.6.5 Segmentation

The actual segmentation of the audio stream into speech/music segments is done by classifying each second of speech into speech or music, and then segmenting the audio where changes in output classes occurs. To avoid single misclassifications a simple rule is applied to the classification output. The rule transforms a sequence of `XXYXX` to `XXXXX`, where `X` is 1 second of audio classified as either music or speech, `Y` then represent the opposite class. This output rule is applied to avoid too many 1 seconds segments.

An example of a segmentation of a $182s$ long audio stream is shown in figure 3.17(a). The



(a)



(b)

Figure 3.17: The upper plot of (a) shows the segmentation of our system on a $182s$ long audio stream. The lower part show the true segmentation of audio stream. The system makes two misclassifications. (b) shows the audio signal where the errors occurs, and it can be seen that the misclassifications are due to two silence periods in the music.

audio stream contains speech from $0-56s$, music from $56-113s$, and speech from $113-182s$. The estimated class from the system is shown in the upper plot and the true class is shown in the lower plot. The system makes two misclassifications at $66-67s$ and $68-69s$. In figure 3.17(b) the audio signal from $66-70s$ is shown. The figure shows that the audio signal contains two silence periods, which are causing the misclassifications. Note that the output rule does not work in this case, as the output sequence is: `MMSMSMM`, where `M` is music and `S` is speech.

3.7 Discussion

The results of the work on speech/music classification showed that very good results were obtainable using a feature set consisting of a limited number of features.

The overall test misclassification rate was 2.2% for the linear classifier using the 14 features proposed by the backward elimination scheme. This is comparable to the results reported by for instance [Li et al., 2001] using MFCCs, and comparable to other results reported in the audio classification field.

The good result obtained using the simple linear discriminant and the observation that many combinations of features giving more or less similar results, suggest that the classes are well separated in feature space, as was also indicated by [Scheirer and Slaney, 1997].

Among the features that were selected in the feature selection schemes, there is a tendency that the features, measure the energy of signals, are preferred. The fact that speech in general contains less energy than music governs our model somewhat. If more classes such as environment sounds, noise, speech over music etc. were considered, energy-based features may not be sufficient. If more audio classes where considered, other features and more complex classifier may be needed.

The choice to classify based on 1-second windows does yield a very high correct classification rate. Decisions would be even clearer if longer windows were considered. On the other hand longer decision windows would make it harder to locate the exact time of changes because of the lower resolution.

3.8 Summary

In this chapter four different classifiers were used to classify audio into speech and music. We saw that the classifiers performed almost equally well using the 60 proposed features extracted on a 1 second basis. Then we used two approaches to decrease the dimensionality of the feature space. The first approach was based on pruning a NN. We saw that the number of features could be decreased without loss in the classification performance. We also showed that different feature combinations where able to obtain the same performance. The second approach to decrease the number of features was based on backward elimination and forward selection of the linear discriminant. Also this simple classifier was able to obtain very good results with a low number of features. Finally the linear discriminant and 14 features was selected as the final classification model, and the model was evaluated.

Chapter 4

Speaker Change Detection

The speech/music classification gives an initial partitioning of the audio stream. Observing a typical news broadcast, shows that different stories are sometimes separated using a jingle but most commonly only indicated by speaker changes. Using only music to separate speech segments could result in long segments with multiple unrelated stories. Thus, by finding speaker changes in the audio will make a transcription easier to inspect.

Speaker change detection is an aspect of event detection which in the case of audio streams concerns finding notable changes in the stream, namely changes in channel conditions, environment and speaker changes. This segmentation of the audio stream into homogenous speaker segments is a research topic that has been widely studied over the last years. The indexing task has an impact on the performance of speaker dependent speech recognizers as it allows the system to adapt its acoustic model to the given speaker and/or environment and thereby improve recognition performance.

Speaker change detection approaches can roughly be divided into three classes: energy-based, metric-based and model-based methods.

Energy-based methods rely on thresholds on the audio signal energy, placing changes at 'silence' events or by considering sudden changes in energy levels. This approach could be used to give an initial segmentation that could be refined by more elaborate methods. In news broadcasts we have observed that the audio production can be quite aggressive, with only little if any silence between speakers, making this approach less attractive.

Metric-based methods basically measure the difference between two consecutive frames that are shifted along the audio signal. A number of distance measures have been investigated such as the symmetric Kullback-Leibler distance [Siegler et al., 1997]. Parametric models have also been deployed for instance using a likelihood ratio to perform statistical hypothesis tests [Kemp et al., 2000]. Parametric models corrected for finite samples using the

Bayesian Information Criterion (BIC) are also widely used which is thoroughly investigated in [Cettolo et al., 2005]. The BIC procedures proposed are computationally heavy which has resulted in a number optimizations. Huang and Hansen [Huang and Hansen, 2004a] argued that BIC-based segmentation works well for longer segments, while a BIC approach with a preprocessing step that uses a T^2-statistic to identify potential changes, was superior for short segments and reduced the computational load.

Nakagawa and Mori [Nakagawa and Mori, 2003] compare different methods for speaker change detection, including Generalized Likelihood Ratio, BIC, and a vector quantization based (VQ) distortion measure. The comparison indicates that the VQ method is superior to the other methods. A simplification of the Kullback-Leibler distance, the so-called divergence shape distance (DSD), was presented in [Lu and Zhang, 2005] for a real-time implementation. The system includes a method for removing false alarms using "lightweight" GMM speaker models.

Model-based methods are based on recognizing specific known audio objects, e.g., speakers, and classify the audio stream accordingly like the approach we used for speech/music discrimination in chapter 3. This approach has been very successful for event detection if the audio classes are well-defined, for instance in separating male and female speakers where representative training data may be obtained. The model-based approach has been combined with the metric-based to obtain hybrid-methods that do not need prior data [Kemp et al., 2000, Kim et al., 2005].

In general performance of unsupervised speaker change detection methods reported recall[1] (RCL) 79.4 % and precision (PRC) 78.9 % for the T^2-BIC algorithm in [Huang and Hansen, 2004a]. The DSD-metric used in Lu and Zhang [2005] resulted in RCL/PRC of 89%/85% for long speaker segments (longer than 3 seconds). The results are obtained by applying false alarm compensation schemes.

One of the requirements to our system is that it should work with no prior information on speaker identity. This also means that the speaker indexing system must be unsupervised, with no prior information on speaker identities, number of speakers and so on. Since we are interested in segmenting news with an unknown group of speakers we limit our investigation to metric-based methods. Furthermore we are interested in a system that is not too specialized to a given channel, hence, in both system design and in the evaluation procedure we will focus on the issue of robustness.

This chapter presents our investigation of speaker change detection. The presentation will present the features, followed by an introduction of metric-based change detection and the distance measures used. We then present how an algorithm for the change detection is built. The evaluation is done on a database that will be presented in conjunction with the evaluation of the algorithms.

Our focus has been on applying the vector quantization distortion metric to the change detection. In addition we have implemented two other widely used metrics to compare the performance.

[1]Recall is defined as the number of correctly found change-points divided by the total number of true change-points. Precision is the number of correctly found change-points divided by the number of hypothesized change-poins.

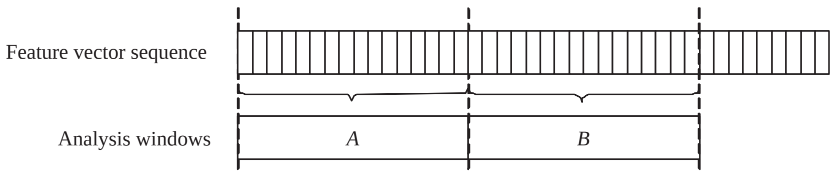


Figure 4.1: The feature vector sequence is divided into two succeeding analysis windows A and B.

The work in speaker change detection has been reported in the paper UNSUPERVISED SPEAKER CHANGE DETECTION FOR BROADCAST NEWS SEGMENTATION, which is included in appendix A. Some changes in the notation have been inferred, though.

4.1 Feature Selection

The features proposed for event detection are numerous. Different features have been introduced, depending on the application. As the intention is to separate different speakers, it has been natural to look at features used for speaker recognition. MFCCs have proven to be very useful for this task, see e.g., [Ganchev et al., 2005] and our investigation of speaker recogniton [Jørgensen and Mølgaard, 2005]. Therefore the MFCCs have been chosen for the features for the speaker change detection, and other features are not considered.

The use of MFCCs moreover saves overall computation time because the values calculated can be reused.

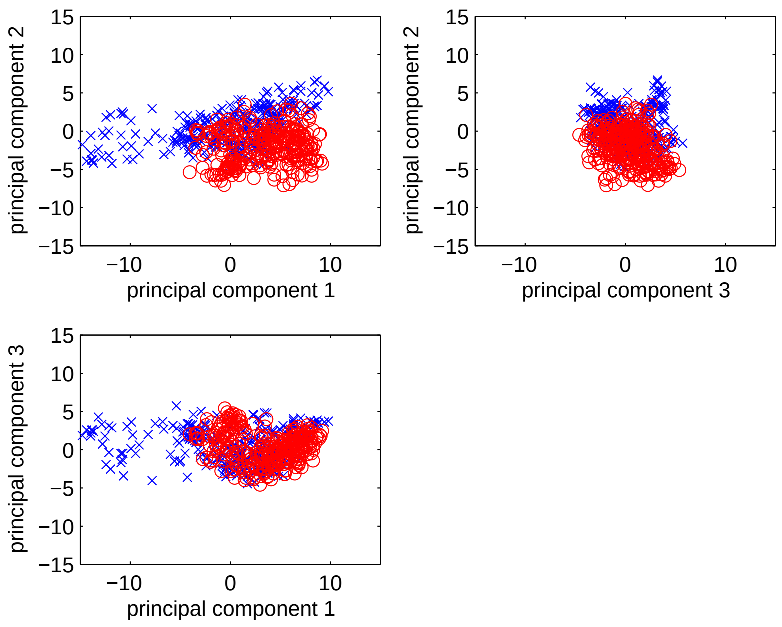
4.2 Distance Measures

Basically change detection tries to locate significant changes in the audio signal. Metric-based change detection is done by compare two successive analysis windows that are shifted along the signal. The distance indicates the similarity between the two windows, so a peak in the metric may indicate a change in the sound stream.

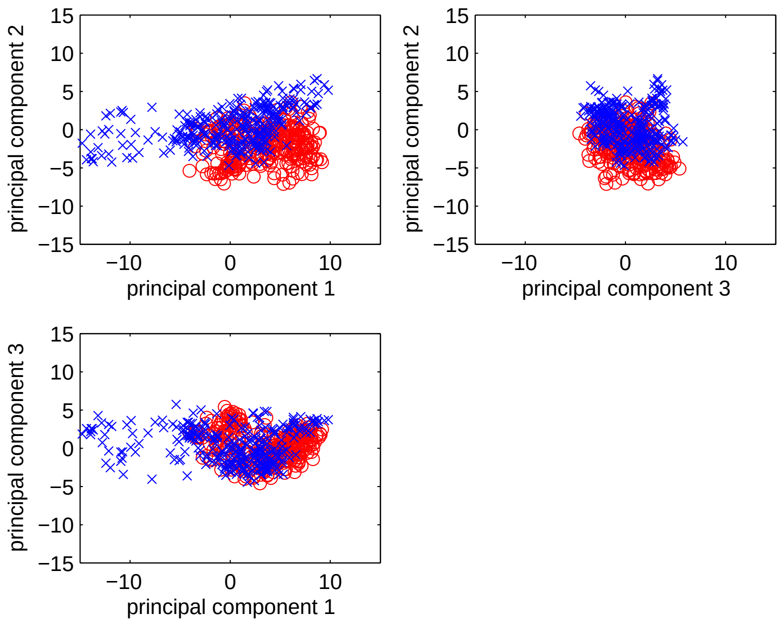
We define A and B to be two succeeding sequences of feature vectors of length N: $A = \{\mathbf{x}_1^A, \mathbf{x}_2^A, ..., \mathbf{x}_N^A\}$ and $B = \{\mathbf{x}_1^B, \mathbf{x}_2^B, ..., \mathbf{x}_N^B\}$, see figure 4.1.

Figure 4.2 shows a PCA-plot of such two succeeding feature vector sequences, where a speaker change-point occurs in between the two sequences. Each sequence is three seconds long. The audio is taken from a CNN broadcast news show, and the feature vector consist of 12 MFCCs. The figures show quite a large overlap in the feature vector distributions of the speakers, but one of them seems to have a larger variance along the 1st principal component.

Below we present three different distance measures that have been considered in this context. The basic approach of the three methods is to summarize the features in each window, and then infer a metric that can be evaluated.



(a) Plot with blue segment at back



(b) Plot with red segment at back

Figure 4.2: PCA-plot of 3 seconds of speech from two different speakers in a CNN news broadcast show. Crosses and circles marks the cluster centers from speaker one and speaker two respectively. (a) and (b) show the same plot the only difference is the order in which the feature vectors are plotted.

4.2.1 Vector Quantization

The vector quantization distortion measure is based on clustering the feature vector sequence into K centroids. In [Kinnunen et al., 2000] five different clustering algorithms are compared for use in a speaker identification setup using vector quantization. The comparison shows only marginal difference in the results obtained with the five algorithms. Thus, in this project we have chosen the K-means clustering algorithm, see e.g., [Bishop, 1995].

K-means Clustering Algorithm

K-means works by assigning each feature vector in the sequence $A = \{\mathbf{x}_1^A, \mathbf{x}_2^A, ..., \mathbf{x}_N^A\}$ to one of K centroids $\mathbf{C} = \{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_K\}$ in a way to minimize the sum-of-squares error function:

$$E = \sum_{j=1}^{K} \sum_{n \in S_j} ||\mathbf{x}_n^A - \mathbf{c}_j||^2, \tag{4.1}$$

Where S_i is the set of feature vectors assigned to centroid $\mathbf{c}_i$. The centroid is simply calculated as the mean of the feature vectors assigned to that centroid.

$$\mathbf{c}_j = \frac{1}{N_j} \sum_{n \in S_j} \mathbf{x}_n^A \tag{4.2}$$

N_j is the number of data points in S_j.

The K-means algorithm begins by choosing K random feature vectors in A as initial centroid. Each feature vector in A is assigned to the nearest center, and each centroid is re-computed using equation (4.2). Each feature vector is then re-assigned to its new nearest centroid. This procedure is repeated until no more changes occur in the assignment. Each cluster centroid $\mathbf{c}_j$ is called a code-vector and the set of code-vectors $\mathbf{C}$ is called a codebook.

Vector Quantization Distortion

The vector quantization based metric is based on clustering the analysis windows. The windows are then compared using the so-called VQ-distortion measure VQD between the features in window B and the codebook $\mathbf{C}^A$ calculated on A, as defined in [Nakagawa and Mori, 2003]:

$$\text{VQD}(\mathbf{C}^A, B) = \frac{1}{N} \sum_{n=1}^{N} \underset{1 \le k \le K}{\arg\min} \left\{ d\big(\mathbf{c}_k^A, \mathbf{x}_n^B\big) \right\}, \tag{4.3}$$

Where $\mathbf{c}_k^A$ denotes the k-th code-vector in $\mathbf{C}^A$, $1 \le k \le K$. d is a distance between two feature vectors. In this context the well-known *Euclidean*[2] distance d_E is used. Thus, the VQD measure is the mean distances from all the feature vectors $\mathbf{x}_n^B$ in B to the nearest code-vector $\mathbf{c}_k^A$ in $\mathbf{C}^A$.

[2]For reference: $d_E(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$

4.2.2 Kullback-Leibler Distance

The VQD approach models each window using clustering. The Kullback-Leibler distance approach is based on estimating whether the density functions of the feature distributions are equal.

The symmetric Kullback-Leibler distance (KL2) has been used in speaker identification systems and applied to general change detection [Meinedo and Neto, 2003]. The KL2 distance between two analysis windows A and B is defined as:

$$\mathrm{KL2}(A,B) = \int_{\mathbf{x}} [p_A(\mathbf{x}) - p_B(\mathbf{x})] \log \frac{p_A(\mathbf{x})}{p_B(\mathbf{x})} d\mathbf{x} \tag{4.4}$$

Assuming that the feature sequences A and B are n-variate Gaussian distributed, $p_A \sim \mathcal{N}(\boldsymbol{\mu}_A, \boldsymbol{\Sigma}_A)$, $p_B \sim \mathcal{N}(\boldsymbol{\mu}_B, \boldsymbol{\Sigma}_B)$, i.e.

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\Big\{ -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\Big\} \tag{4.5}$$

Combining equations (4.4) and (4.5) (details are given in appendix C) yields the following expression:

$$\begin{aligned}\mathrm{KL2}(S^A, S^B) &= \frac{1}{2}\mathrm{Tr}\Big[(\boldsymbol{\Sigma}_A - \boldsymbol{\Sigma}_B)(\boldsymbol{\Sigma}_B^{-1} - \boldsymbol{\Sigma}_A^{-1})\Big] + \\ &\quad \frac{1}{2}\mathrm{Tr}\Big[(\boldsymbol{\Sigma}_A^{-1} + \boldsymbol{\Sigma}_B^{-1})(\boldsymbol{\mu}_A - \boldsymbol{\mu}_B)(\boldsymbol{\mu}_A - \boldsymbol{\mu}_B)^\top\Big] \end{aligned} \tag{4.6}$$

This expression is called the *divergence* for two gaussian distributed classes.

4.2.3 Divergence Shape Distance

The KL2 distance presented above is composed of two terms, the former is based solely on differences between the covariance matrices and the latter involves differences between the mean vectors. These components can be characterized as differences in shape and size of the probability density functions, respectively. The mean vectors, included in the latter term, can vary much depending on the environment [Lu and Zhang, 2005], which may be reflected in the metric even though actual speaker changes are not present. Using only the first term should remove this dependency, so that only the difference between covariances contribute. This metric is called the divergence shape distance (DSD).

$$\mathrm{DSD}(A,B) = \frac{1}{2}\mathrm{Tr}\Big[(\boldsymbol{\Sigma}_A - \boldsymbol{\Sigma}_B)(\boldsymbol{\Sigma}_B^{-1} - \boldsymbol{\Sigma}_A^{-1})\Big] \tag{4.7}$$

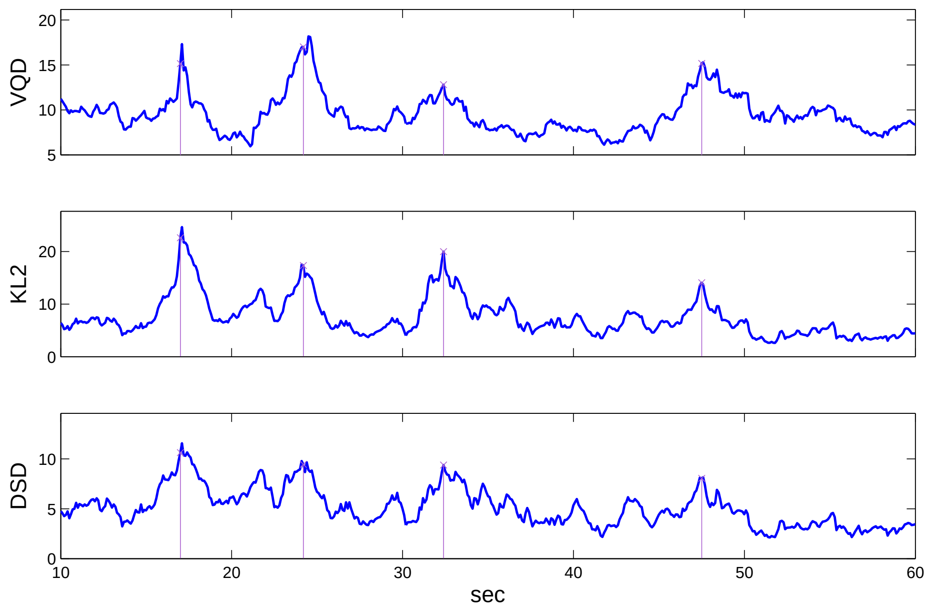


Figure 4.3: The three metrics VQD, KL2, and DSD are calculated on a CNN broadcast news show. The show contains four speaker change-point indicated by the vertical lines.

4.2.4 Example

In the sections above three distance metrics were introduced, namely the VQD, KL2, and DSD. In the three presented distance metrics a greater value means a greater difference in the two distributions. In figure 4.3 the three metrics are calculated on CNN broadcast news show. The sample contains four speaker change-points indicated by the vertical lines. All three metrics gives peaks at the true change-points. Though, not all change-points are equally well indicated, and peaks aside from the true change-points are seen. The DSD metric shows the largest ambiguity in the distance-values.

4.3 Change Detection Algorithm

Based upon the calculated distance metric the change detection algorithm determines whether or not a speaker change occurred. Our algorithm works in two steps. The first step is the change-point detection where candidate change-points are found. The second step is the false alarm compensation.

The speaker change detection algorithm was developed using the VQ approach. This section

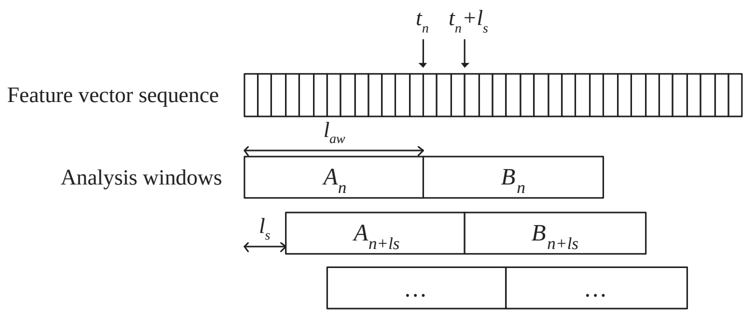


Figure 4.4: The feature vector sequence is divided into analysis windows. A_n is the analysis window with length l_{aw} and endtime t_n. B_n is the succeeding analysis window with starttime t_n and length l_{aw}. The analysis windows are shifted with l_s along the feature vector sequence.

first presents how the VQD distance metric are calculated. Then we describe how the change-point detection algorithm finds potential change-points. Next we propose a false alarm compensation algorithm, that is used to reject some of the wrongly detected change-points. We then give some considerations about how the parameters of the algorithm should be adjusted. Finally we present an example, where we demonstrate how our algorithm is used to first find candidate change-point and afterwards the false alarm compensation is rejecting the false alarms.

4.3.1 Vector Quantization Distortion Measure Calculation

The feature vector sequence is divided into analysis windows of length l_{aw}. Let A_n be the analysis window with endtime t_n and B_n the succeeding analysis window with starttime t_n. The analysis windows are shifted with length l_s, thus the shifted analysis windows at time $t_n + l_s$ are called A_{n+l_s} and B_{n+l_s}. See figure 4.4 for a graphical illustration.

As described above, the conventional VQ-algorithm computes the distortion measure between two feature vector sequences A_n and B_n by computing $\mathrm{VQD}(\mathbf{C}_n^A, B)$, clustering A_n, to obtain the codebook $\mathbf{C}_n^A$. We have considered two optimizations for this VQ-algorithm.

The first improvement is done by using the codebook $\mathbf{C}_n^B$ of B instead of comparing with the whole sequence of feature vectors. Thus, the metric used at time t_n in the change point detection is instead $\mathrm{VQD_n} = \mathrm{VQD}(\mathbf{C}_n^A, \mathbf{C}_n^B)$. This optimization should save a number of comparisons, as the number of vectors is reduced to the number of code vectors used in the codebook.

The second optimization concerns the K-means algorithm. The clustering is by far the most time consuming operation in the overall change detection method. It is observed that there is a large overlap of A_n and the next window A_{n+l_s}. It is therefore very convenient to use the cluster centers obtained for A_n as initial centers for A_{n+l_s}. This makes the K-means algorithm converge faster and minimizes the distance between two succeeding codebooks, resulting in less fluctuating distortion measures.

The implementation of the k-means clustering algorithm was done using the voicebox toolbox [Brookes, 1998]

4.3.2 Change-point Detection

Figure 4.3 showed the three distance measures for a sample file. Based on this metric the change-point detection algorithm should find the speaker changes. We have defined a simple change-point detection algorithm. The basic algorithm evaluates the calculated distance metric VQD_n at every time step time t_n. A change-point is found if VQD_n is larger than a threshold $\mathrm{th}_{\mathrm{cd}}$. The figure 4.3 shows that near the true change-points there are typically more than one local peak, thus to detect only the true change-point VQD_n should be the local peak within T_i seconds.

This procedure is called the changePointDetection and the pseudo-code for the algorithm can be seen in **Algorithm 4**. This algorithm works in three steps. Firstly codebooks for every time step t_n are calculated using K-means. The K-means uses $\mathbf{C}_{n-l_s}$ as initial cluster centroids when $\mathbf{C}_n$ are calculated. Next are the VQD_n measure found for each time step. Note that $\mathbf{C}_n$ and $\mathbf{C}_{n+l_{aw}}$ are two succeeding feature vector sequences corresponding to A_n and B_n in figure 4.4 respectively. Finally the change-points are found. A change-point to time t_n is detected if VQD_n is larger than the threshold $\mathrm{th}_{\mathrm{cd}}$ and is the largest VQD values within T_i seconds.

4.3.3 False Alarm Compensation

When running the changePointDetection algorithm it is necessary to keep the analysis window relatively short in order to be able to detect short speaker segments. On the other hand short analysis windows may lack data to make fully reliable models, which consequently may cause false alarms.

The baseline approach yields a number of potential change-points, dividing the audio stream into speaker segments. The generated speaker segments can be used to make more accurate models, because of the larger amount of data. Comparing these more accurate models can then be used to accept or reject the potential change-point.

The false alarm compensation algorithm simply works by making two speaker VQ-codebooks, for the speaker segment before the change-point $\mathbf{C}_{\mathrm{before}}$ and another after the change-point $\mathbf{C}_{\mathrm{after}}$.

The two VQ-distortion measures $\mathrm{VQD}(\mathbf{C}_{\mathrm{before}}, \mathbf{C}_{\mathrm{after}})$ and $\mathrm{VQD}(\mathbf{C}_{\mathrm{after}}, \mathbf{C}_{\mathrm{before}})$ are computed and the mean $\mathrm{VQD}_{\mathrm{mean}}$ of these two measures is found. The change-point is then accepted if the measure is larger than the threshold $\mathrm{th}_{\mathrm{fac}}$ and rejected if it is below. We found that using the mean of the two distortion measures is more stable than using just one of the measures.

If a real speaker change is missed during the initial change-point detection, the resulting speaker model would contain data from two speakers, meaning that the speaker codebook

Algorithm 4 changePointDetection

```
$\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\} \leftarrow$feature vector sequence

// Calculate codebooks
$n \leftarrow l_{aw}$
while $n < N$ do
    $A_n \leftarrow \{\mathbf{x}_{n-l_{aw}+1}, ..., \mathbf{x}_n\}$
    $\mathbf{C}_n \leftarrow$ K-means$(A_n)$
    $n \leftarrow n + l_s$
end while

// Calculate distortion measure
$n \leftarrow l_{aw}$
while $n < N$ do
    $\mathrm{VQD_n} \leftarrow \mathrm{VQD}(\mathbf{C}_\mathrm{n}, \mathbf{C}_{\mathrm{n+l_{aw}}})$
    $n \leftarrow n + l_s$
end while

// Find change-points
$n \leftarrow l_{aw}$
while $n < N$ do
    if $\mathrm{VQD_n} > \mathrm{th_{cd}}$ then
        if $\mathrm{VQD_n} == \max\left(\{\mathrm{VQD_{n-T_{max}}}, ..., \mathrm{VQD_{n+T_{max}}}\}\right)$ then
            $\mathrm{CP_n} \leftarrow$ TRUE
        else
            $\mathrm{CP_n} \leftarrow$ FALSE
        end if
    end if
    $n \leftarrow n + l_s$
end while

return $\mathrm{CP_n}$
```

models both speakers. To counteract this problem only the $T_{\max}$ seconds closest to the change-point are used to make the speaker codebook.

4.3.4 Parameter Settings

The proposed change-point detection algorithm requires some parameters to be adjusted. The two thresholds $\mathrm{th_{cd}}$ and $\mathrm{th_{fac}}$ should be set to optimize the desired relation between finding all the true changes and inferring false alarms. As proposed by [Lu and Zhang, 2005] an automatic threshold adjustment method is used.

We use $\mathrm{VQD_{n,avr}}$ as the average of the distortion measure in a window of $2T_{\max}$ around t_n:

$$\mathrm{VQD_{n,avr}} = \frac{1}{2T_{max} + 1} \sum_i \mathrm{VQD_{n+i}} \tag{4.8}$$

Algorithm 5 FALSEALARMCOMPENSATION

```
$\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\} \leftarrow$feature vector sequence
$\{A_{\mathrm{CP}_1}, A_{\mathrm{CP}_2}, ..., A_{\mathrm{CP}_\mathrm{N}}\} \leftarrow$ get speaker segments
$\mathrm{CP_n} \leftarrow$ get potential change-points
$n \leftarrow l_{aw}$

// Iterate all potential change-points
while $n < N$ do
    if $\mathrm{CP_n} ==$ TRUE then

        // Find speaker segment before change-point
        if length$(A_{\mathrm{CP_n}}) > T_{\mathrm{max}}$ then
            $A_{\mathrm{before}} \leftarrow \{\mathbf{x}_{n-T_{max}}, ..., \mathbf{x}_n\}$
        else
            $A_{\mathrm{before}} \leftarrow A_{\mathrm{CP_n}}$
        end if

        // Find speaker segment after change-point
        if length$(A_{\mathrm{CP_{n+1}}}) > T_{\mathrm{max}}$ then
            $A_{\mathrm{after}} \leftarrow \{\mathbf{x}_{n+l_s}, ..., \mathbf{x}_{n+T_{max}}\}$
        else
            $A_{\mathrm{after}} \leftarrow A_{\mathrm{CP_{n+1}}}$
        end if

        // Calculate codebooks
        $\mathbf{C}_{\mathrm{before}} \leftarrow$ K-means$(A_{\mathrm{before}})$
        $\mathbf{C}_{\mathrm{after}} \leftarrow$ K-means$(A_{\mathrm{after}})$

        // Calculate $VQD_{mean}$
        $\mathrm{VQD_{mean}} \leftarrow \frac{1}{2}\Big(\mathrm{VQD}(\mathbf{C_{before}}, \mathbf{C_{after}}) + \mathrm{VQD}(\mathbf{C_{after}}, \mathbf{C_{before}})\Big)$

        // Accept or reject change-point
        if $\mathrm{VQD_{mean}} > \mathrm{th_{fac}}$ then
            $\mathrm{CP_n} \leftarrow$ TRUE
        else
            $\mathrm{CP_n} \leftarrow$ FALSE
        end if
    end if
    $n \leftarrow n + l_s$
end while

return $\mathrm{CP_n}$
```

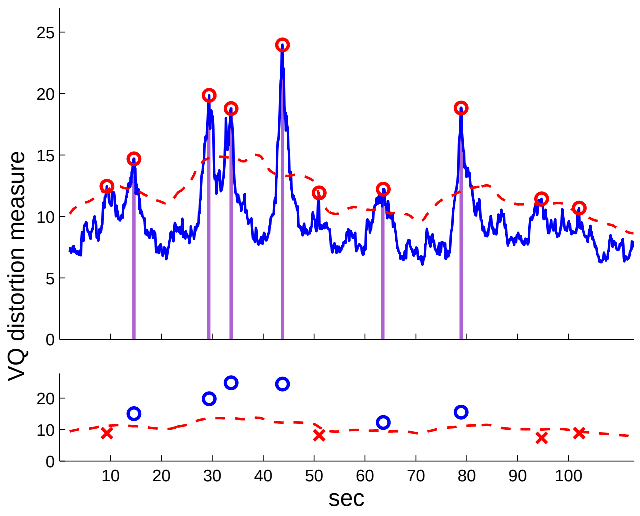


Figure 4.5: The upper part of the figure shows the VQ-distortion measure $\mathrm{VQD_n}$ for a sample file. The true speaker changes are indicated by vertical lines. The dotted line indicates the threshold $\mathrm{th_{cd}}$ and the estimated change-points found are shown with circles. In addition to the true speaker change-points four false change-points are found. The lower part of the figure shows the VQ-distortion $\mathrm{VQD_{mean}}$ for the found change-points. The threshold $\mathrm{th_{fac}}$ is indicated and the accepted change-points are shown by circles, and the rejected are shown by crosses.

Where $-T_{\max}/l_s < i < T_{\max}/l_s$. The thresholds at time t_n are thereby set to:

$$\mathrm{th_{cd,n}} = \alpha_{\mathrm{cd}} \mathrm{VQD_{n,avr}} \tag{4.9}$$

$$\mathrm{th_{fac,n}} = \alpha_{\mathrm{fac}} \mathrm{VQD_{n,avr}} \tag{4.10}$$

The two amplifiers α_{cd} and α_{fac} must be set in advance.

The timing parameters l_{aw}, T_i, and $T_{\max}$ should be set according to the expected distribution of speaker turn lengths. l_s defines the resolution of the detected change-points.

4.3.5 Example

An example of the change-point detection algorithm is shown in figure 4.5. The audio clip in this example is 113 seconds long and contains speaker change-points at 14.6, 29.3, 33.7, 43.8, 63.5, and 78.9 seconds indicated by the vertical lines. The upper part of the figure shows the VQ-distortion measure $\mathrm{VQD_n}$ as function of time. The dotted line indicate the threshold $\mathrm{th_{cd}}$ and the estimated change-points found by our change-point algorithm are shown with circles. It is seen that in addition to the true speaker change-points four false false alarms occur.

The lower part of the figure shows the VQ-distortion measure $\mathrm{VQD}_{\mathrm{mean}}$ for the found change-

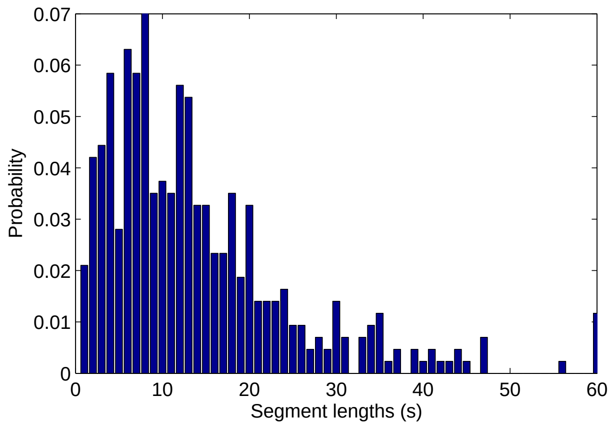


Figure 4.6: Histogram of the speaker segment lengths contained in the database

	Total length (min)	Avg. segment length (sec)	Speaker changes
CNN	38	17.0	134
CBS	20	9.9	121
WNYC	26	22.6	69
PRI	19	15.8	64
All	103	15.6	388

Table 4.1: Database information

points. Again, the dotted line indicate the threshold th_{fac} and the accepted change-points are shown by circles, and the rejected are shown by crosses.

In this example all the true speaker changes are found, and false alarms are removed by the false alarm compensation step.

4.4 Experiments and Results

In this section we will evaluate the various VQD parameters. Next we will compare the VQD measure with the two other measures KL2 and DSD. The generalizability of the change detection algorithm using the VQD measure is then evaluated.

4.4.1 Speech Database

The speech data used was news-podcasts obtained from four different news/radio channels CNN, CBS, WNYC, and PRI.

The data consists of 103 min of broadcast news, which contains speech from numerous speakers in different environments. The database only contains speech segments as we are only interested in evaluating the *speaker* change detection performance.

The length of the speaker segments range from 0.4 seconds to 119 seconds with an average length of approximately 14 seconds. Figure 4.6 shows the distribution of the segment lengths. The number of speaker changes is 388, distributed over 47 files. The data was manually labelled into different speakers. The total number of segments is 435, and 75 of these have a length less than 5 seconds. These segments are generally considered difficult to detect [Lu and Zhang, 2005, Huang and Hansen, 2004a].

4.4.2 Evaluation Measures

A change-point proposed by the algorithm may not be precisely aligned with the manual label. This occurs for instance if the change occurs at a silence period or if speakers interrupt each other. To take this into account, a found change is counted as correct if it is within 1 second of the manual labelled change-point as in [Huang and Hansen, 2004a]. The *mismatch* is defined as the time between a correct found change-point point and the manually labelled one.

Two kinds of errors are possible in this context. A missed change-point and inferring a false alarm. These errors are also referred to as deletions and insertions. The evaluation measures frequently used are recall (RCL) and precision (PRC):

$$\text{RCL} = \frac{\textit{no. of correctly found change-points}}{\textit{no. of true change-points}} \tag{4.11}$$

$$\tag{4.12}$$

$$\text{PRC} = \frac{\textit{no. of correctly found change-points}}{\textit{no. of hypothesized change-points}} \tag{4.13}$$

The F-measure combines RCL and PRC into one measure,

$$\text{F} = \frac{\text{RCL} \times \text{PRC}}{\alpha \times \text{RCL} + (1-\alpha)\text{PRC}} \tag{4.14}$$

with α as a weighting parameter that can be used to emphasize either of the two quantities. The results presented below use the equal weighting, with $\alpha = 0.5$.

4.4.3 Parameter Tuning

The speaker change detection algorithm requires some parameters to be set. In this section we will investigate the parameters influence on the change detection performance.

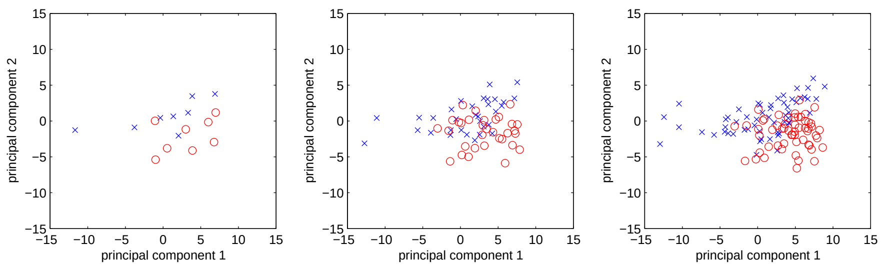


Figure 4.7: The figure shows how the two feature vector sequences from figure 4.2 are clustered using 8, 32, and 56 clusters. Crosses and circles represent the cluster centers from the two sequences (speakers) respectively.

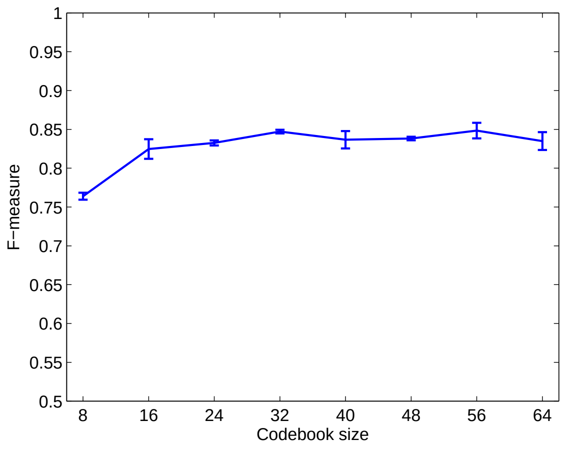


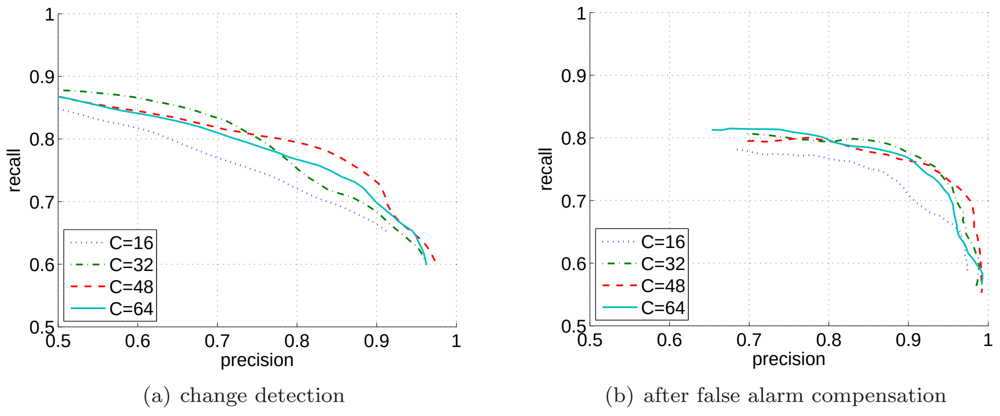
Figure 4.8: F-measure as function of the number of clusters used when creating the codebooks. The figure shows the mean and standard deviation from 3 runs of each codebook size.

Codebook Size

The VQD measure is based on clustering the feature vectors into K disjoint clusters. The number of clusters K, which also defines the codebook size, must be set in advance. In [Kinnunen et al., 2000] the performance dependency of the codebook size are investigated in a speaker identification setup using vector quantization. The conclusion of this investigation is that the identification rate is increased when increasing the codebook size. The only drawback of increasing the codebook size, is an increase in the runtime of the algorithm.

Figure 4.7 shows the cluster centers of clustering the feature vector sequences from figure 4.2 using 8, 32, and 56 clusters respectively.

Figure 4.8 shows the F-measure as function of codebook size in the VQD measure. The codebook size has been varied in both the change-point detection and the false alarm compensation scheme. The figure shows mean and standard deviation of three runs for each



(a) change detection (b) after false alarm compensation

Figure 4.9: RCL-PRC-curve for different choices of codebook size. (a) shows the curve before the false alarm compensation, (b) shows the curve after the false alarm compensation. The figures are created by varying the α_{cd}, in (b) α_{fac} is kept constant.

codebook size using all the data from our speech database. The threshold parameters α_{cd} and α_{fac} has been optimized to obtain the highest F-measure in each run. The analysis window length is set to 3 seconds, which means that each analysis window contains 300 feature vectors. The figure shows the highest F-measure using 32 clusters. Further increase in codebook size does not increase the F-measure further.

The F-measure is a good indication of the performance of the algorithm, but it only tells something about one specific choice of the threshold parameter. By varying the thresholds a RCL-PRC-curve can be created. Figure 4.9 shows this RCL-PRC-curve for different choices of codebook sizes. The figure shows, similar to figure 4.8, that there is not a single best choice of the codebook size, though it should be chosen to be 32 or beyond.

Analysis Window Length

The length of the analysis window has a great impact on the change detection performance. The analysis window should be large enough to contain enough data to estimate the segment models. Though, if a too large window length is chosen, the window could contain speech from multiple speakers. This could result in that a change-point could be missed.

Figure 4.10 shows the F-measure as function of the length of the analysis window. The threshold parameters α_{cd} and α_{fac} has been optimized to obtain the highest F-measure in each run. The figure shows that the optimal F-measure is obtained when setting the analysis window to 3 seconds.

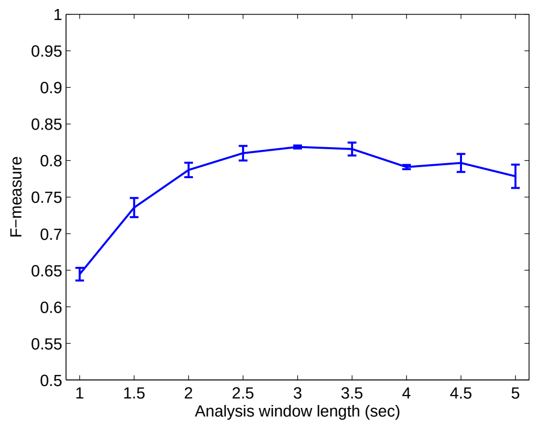


Figure 4.10: F-measure as function of the length of the analysis windows. The figure shows the mean and standard deviation from 3 runs of each length.

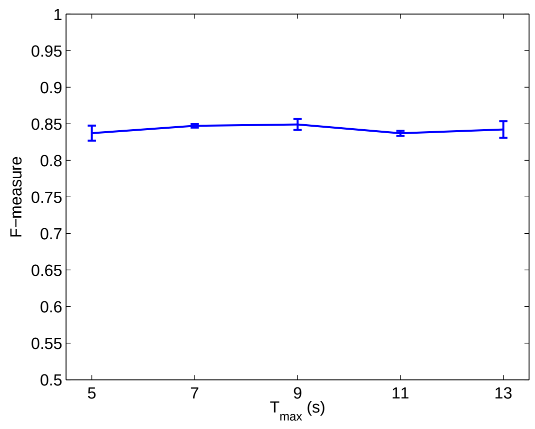


Figure 4.11: F-measure as function of the $\mathrm{T}_{\max}$ parameter.

Maximum Window Size - $T_{\max}$

In the false alarm compensation algorithm only the $T_{\max}$ seconds nearest the potential change-point is used to create the codebooks. Figure 4.11 shows the F-measure obtained by varying $T_{\max}$. The figure shows that the specific choice of this parameter has no great influence of F-measure. Though, not having the $T_{\max}$ constraint the F-measure is decreased to 0.814, which should be compared to the F-measure 0.845 that is obtained when setting $T_{\max} = 9s$. In addition to the improved performance on having the $T_{\max}$ constraint, the runtime is improved, due to the reduced amount of data for each evaluation.

Metric	F	RCL	PRC	Mismatch
VQD32	0.761	0.838	0.697	191ms
VQD32-FAC	0.845	0.763	0.946	188ms
VQD48	0.717	0.840	0.627	208ms
VQD48-FAC	0.839	0.766	0.928	206ms
VQD56	0.687	0.863	0.573	220ms
VQD56-FAC	0.854	0.801	0.915	202ms
KL2	0.763	0.833	0.704	212ms
KL2-FAC	0.823	0.789	0.860	212ms
DSD	0.623	0.766	0.526	308ms
DSD-FAC	0.732	0.665	0.814	288ms

Table 4.2: Results obtained with α_{cd} and α_{fac} adjusted to optimize the F measure after the false alarm compensation (FAC). Both the results before and after the FAC are shown.

4.4.4 Metric Comparison

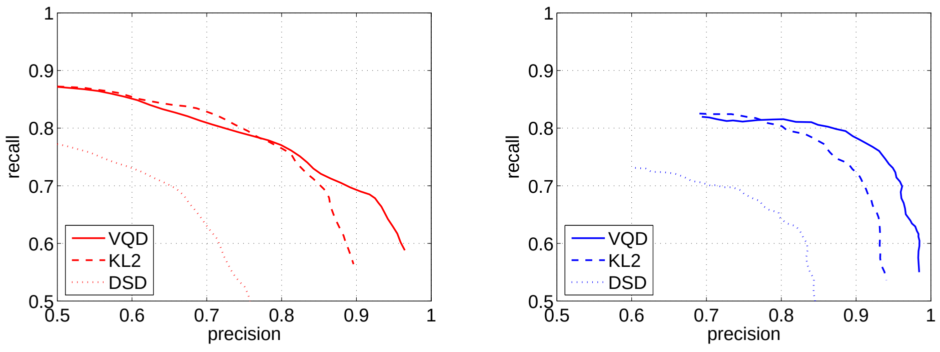
In the past section the optimal settings for the change detection were found. In this section we will compare the three distance metrics VQD, KL2, and DSD.

In the tests the following settings are used: $l_{aw} = 3s$, $T_i = 2s$, $T_{\max} = 8s$, and $l_s = 0.1s$. α_{cd} and α_{fac} are set to maximize the F-measure after the false alarm compensation (FAC). The VQD-approach is evaluated using 32, 48, and 56 clusters for both the change detection and in the false alarm compensation. In the false alarm compensation using the KL2 and DSD approaches, 56 clusters are used.

Table 4.2 shows the results obtained using all the data from our database. This table shows the results both before and after the false alarm compensation (FAC). The best performance is obtained using VQD with 56 clusters, with a F-measure of 0.854. In this case 80.1% of the true change-points are detected with a false alarm rate of 8.5 %. A relative improvement of 59,7% in precision with a relative loss of 7.2% in recall is obtained with our false alarm compensation scheme. The KL2 metric gives a F-measure of 0.823 and DSD gives a F-measure of 0.732.

Varying α_{cd} a recall-precision curve can be created. Figure 4.12(a) shows the RCL-PRC curve for the three metrics VQD-56, KL2, and DSD for the baseline algorithm. The curves for VQD-56 and KL2 are comparable, though VQD-56 gives better precision at lower recall. VQD-56 and KL2 is clearly better than DSD. Figure 4.12(b) shows the RCL-PRC curves after the false alarm compensation. These curves are created by varying α_{cd} and keeping α_{fac} constant. Though, the baseline recall-precision curve for VQD and KL2 is very similar the VQD-FAC performs better than KL2-FAC. A reason for this could be that VQD and KL2 do not locate the same change-points and FAC then rejects more true change-points found by KL2 than found by VQD.

The change-points are found with a relatively small average mismatch of approximately $0.2s$, which is acceptable for most applications.



(a) Change-point detection. (b) After false alarm compensation.

Figure 4.12: RCL-PRC-curve for the three distance metrics VQD, KL2, and DSD. (a) shows the curve for the baseline, and (b) shows the curve after false alarm compensation. The curves are created by varying α_{cd}. In (b) α_{fac} is kept constant.

metric	CD (s)	FAC (s)	total (s)
VQD16	347	141	488
VQD32	916	138	1054
KL2	54	146	200
DSD	51	145	196

Table 4.3: Runtime in seconds to run the change detection algorithm using the different metrics. The audio used to measure the time is $683s$ long and contains 40 speaker changes.

Runtime Analysis

Another important measure to consider when choosing a metric is the runtime of the algorithm. Table 4.3 shows the runtime for the change detection (CD), the false alarm compensation (FAC), and the total runtime using the different distance metrics. The audio used for this test is 683 seconds long and contains 40 speaker change points.

The table shows that using KL2 and DSD are remarkably faster than using VQD. VQD requires the feature vectors to be clustered and this clustering process is a relatively time consuming process. The table also shows that increasing the number of clusters from 16 to 32 increases the time duration from 488 seconds to 1054 seconds.

4.4.5 Generalizability

To investigate the generalizability of our system, another test was set up where the database was divided into a training set and four test sets. The training set contains files randomly chosen from three of the channels, CNN, CBS, and WNYC. Four test sets were created, one for each of the channels including PRI, using the remaining files in the database.

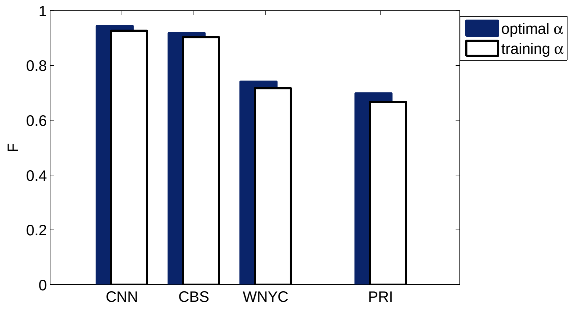


Figure 4.13: This figure shows the results obtained for different test sets. The system optimized for each of the tests are compared with a system optimized for a training set. The figure shows that a threshold chosen on a training set generalize reasonable well to other data sets.

The system was set up using the VQD measure with 56 clusters. The system parameters α_{cd} and α_{fac} were optimized for the training set and then evaluated on the test sets. Figure 4.13 shows the F-measure for this test. The results are compared with the system optimized for each of the specific test sets.

Generally our system performs better on the two test sets CNN and CBS compared to WNYC and PRI. This is most likely due to the fact that WNYC and PRI contain more short segments ($< 3s$) than CNN and CBS. The analysis window length of 3 seonds makes these segments hard to locate.

Only a minor reduction in the F-measure for all test sets is observed when using the training setting compared to the optimal settings for these test sets. Even the data from PRI that was not present in the training set show the same behavior. This demonstrates that the system is robust and lend support to the use in different media without need for further supervised tuning of parameters for new channels.

4.4.6 Analysis of False Alarms and Missed Change-points

Unfortunately not all change-points are detected and despite the false alarm compensation false alarms still occur. In this section we will look into the various reasons for these errors.

Broadcast news speech contains speech in un-ideal conditions. This occurs for instance when reporters report from noisy environments or the anchor speakers speak with background music. A great deal of the false alarms are due to these noisy environments.

As mentioned earlier short speaker segments are difficult to detect. An investigation reveals that approximately 62% of the missed change-points are due to segments that are shorter than 5 seconds.

4.5 Discussion

The results obtained in the speaker change detection compare well to other results presented in the field. The optimal F-measure reported for the VQD approach of 85.4% which is comparable to other state-of-the-art systems, mentioned in the introduction to this chapter.

The VQD approach that we employed was compared to the KL2 and DSD metrics. The performance of the VQD approach was shown to be superior in change-point detection over the two other metrics. One downside of the improved performance is that it comes at a large cost in runtime of the algorithm.

The errors that occur in the segmentation are typically due to short speaker segments. This is a problem that is hard to address using fixed sized windows. Speaker changes are also very difficult to locate if there is a strong constant background sound, such as noise or music. Separating the background noise and speaker could maybe enhance the method. These artifacts however aid the change detection in some cases, as a change in background noise may indicate a change from a studio anchor to a reporter in the field.

4.6 Summary

In this chapter a speaker change detection algorithm based on vector quantization has been developed. The algorithm works in two steps. This first step finds potential change-points, the seconds step either accept or reject the change-points by using more data to estimate the models. We showed that the VQD metric performed better that the two other frequently used measures KL2 and DSD. A F-measure of 0.854 was obtained using all the data in the speech database. We showed that the false alarm rate can be significantly reduced using the false alarm compensation step on the change-points suggested by the first step. We showed that the choice of the thresholds based on one data set generalized reasonably well to other different data sets from different stations.

Chapter 5

Speech Recognition

The final part of the system concerns the transcription of the speech segments. The area of speech recognition has been widely researched for the last decades, which has resulted in many successful systems. Additionally the evaluation of systems has been standardized using a number of speech corpora. Fully investigating the theory and techniques used in modern speech recognition could be the subject of a full thesis, so this description will only review the commonly used methods.

5.1 Introduction

Automatic speech recognition (ASR) systems can be divided into three classes; based on the size of the vocabulary used.

Small vocabulary:
: Used to recognize tens of words. One task is for instance to recognize the digits from 0 to 9. This has for instance been tested using the TIDigits database, which consists of fluent reading of digits. This task can result in more than 99% correct word recognition.

Medium vocabulary:
: Systems recognizing hundreds or few thousands of words. Systems for medium vocabulary tasks can benefit from making speaker dependent models. Systems in this context can achieve a recognition rate of 90-95%.

Large vocabulary:
: The most demanding assignment uses a vocabulary of many thousands words. 60,000 words might for instance be sufficient to recognize the speech in broadcast

news. The large vocabulary automatic speech recognition (LVASR) systems are usually speaker independent, but may still yield a recognition rate of up to 85%.

Several factors contribute to the difficulty of correctly recognizing speech:

Nature of the speech:
: In a dictation system sentences are read with distinct breaks between words. In this way words are clearly separated. Continuous speech on the other hand does not contain clear breaks in between words, making it harder to spot the word changes.

  Continuous speech can be either planned/read or spontaneous. Planned/read speech is normally more clearly pronounced and grammatically correct. Spontaneous speech on the other hand contains incoherent and unfinished sentences.

Environment conditions:
: A noisy environment naturally makes the recognition harder, and therefore research has also been done to counteract noisy speech. Another aspect may be channel conditions, for instance if telephone conversations are considered.

Optimization of the ASR-system therefore depends greatly on the task at hand. Systems that are excellent for one task may very well be infeasible in other contexts. To get satisfactory results, one must therefore take care to choose methodology and parameters of the recognition engine depending on the task at hand.

A number of academical groups have developed ASR-systems, made freely available for instance under the GNU public licence. Popular freely available systems (for academic purposes) include the Hidden markov model ToolKit (HTK) developed at Cambridge University, Sonic [Pellom and Hacioğlu, 2001] from University of Colorado, Torch which was developed at IDIAP in Switzerland and the SPHINX systems originating from Carnegie Mellon University [Walker et al., 2004].

The purpose of this project has not been to develop or improve speech recognition, therefore special optimization or alteration of the speech recognition system has not been considered. Instead a fully developed ASR-system must be considered for the transcription part of our system. The requirements for the ASR-system in our case was therefore:

Large vocabulary:
: The recognition of general broadcast audio requires a large vocabulary.

Speaker independence:
: We have no prior knowledge of speaker identities.

Ease of use:
: The actual ASR implementation is not the focus of this project. Therefore it should be possible to set up the system fairly easily. Additionally we do not have extensive training data, so systems that already are adapted for broadcast news would be preferable.

An initial review of the above systems showed that the general structure of the systems is quite common. This chapter will briefly discuss the structure and some of the theory behind

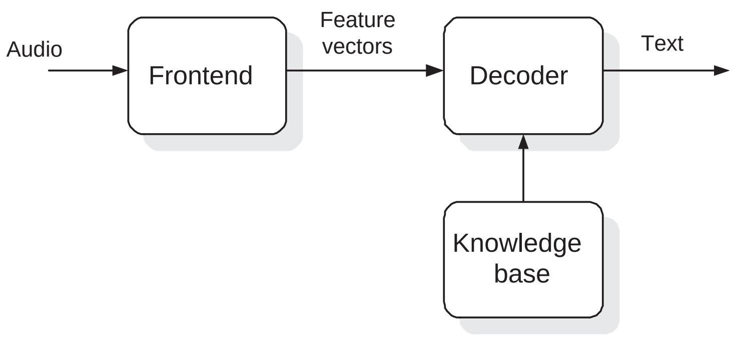


Figure 5.1: General parts of an ASR-system. The frontend does feature extraction. The features are then translated into words by the decoder based on the models in the knowledge base.

common LVASR-systems. This is done to understand how to setup the ASR-system. Finally we present the SPHINX-4-system that was chosen.

5.2 General Speech Recognition

The general concepts and parts of current ASR-systems are commonly accepted among all variants. Generally ASR-systems can be split into the parts shown in figure 5.1. The task of the *frontend* is to transform the raw speech data into features that are convenient for the *decoder* or recognition engine. The decoder then uses models of the speech in the *knowledge base* to map the incoming features onto a sequence of words to obtain the transcript.

Below we present the three parts.

5.2.1 Frontend

The task of the frontend is to do feature extraction, which includes enframing the signal waveform, and transforming the signal to feature vectors used by the decoder. The feature extraction is predominantly some form of cepstral analysis e.g., MFCCs or perceptual linear prediction coefficients (PLP) [Gauvain et al., 2002, Hain et al., 1998]. Features are usually supplemented by first and second order time differences as described in section 2.6.

Features may furthermore be adapted to compensate for noise and channel conditions. Cepstral Mean Normalization (CMN) for instance estimates the long-term average of cepstral feature vectors sequence to subtract it from the computed cepstral features. Vocal Tract Length Normalization (VTLN) estimates a factor that warps the cepstral features to account for variance of the vocal tract of different speakers. This method requires adaption to specific speakers and may therefore not be applicable to the broadcast news task as there is no knowledge of the speakers present in the speech streams.

5.2.2 Knowledge Base

In general the modeling of speech is split into two aspects. The first model is used to map the uttered sounds onto language units. These units may be words, or more predominantly; phonemes.

All words can be split into a sequence of sound units called phonemes. The English language has 40-50 phonemes and thus all words can be described by a combination of these phonemes. The *acoustic model* maps features to phonemes.

The choice of phonemes is guided by using a model of the linguistic structure of the language. This model is called the *language model*. The language model is based on observing the sequence of words so it is usually necessary to employ a *dictionary* to map the phoneme sequence to words. The connection between the three models is illustrated in figure 5.2.

Acoustic Model

The acoustic model attempts to map sounds to a textual representation, which could for instance be words, however, in the case of LVASR, there are simply too many words to be modeled in this way. To create speaker-independent models for each word it is necessary to obtain several samples of every word from several different speakers. Furthermore, the process must be repeated for each new word that is added to the vocabulary, which is why phonemes are preferred.

One problem of phonemes is that the pronunciation of one phoneme is influenced by the neighboring phonemes. For example, the `/AE/`[1] phoneme in the word 'man' sounds different from that in 'lack'. Therefore almost all ASR-systems use triphone or context-dependent phoneme models. Triphone model combinations of phonemes, for instance there is a model for `/M/`→`/AE/`→`/N/` and one for `/L/`→`/AE/`→`/K/`. If a system uses 40 phonemes, it means that there are $40^3 = 64000$ possible triphones, though.

In present LVASR-systems triphones are modeled using Hidden Markov Models (HMM), see e.g., [Pellom and Hacioğlu, 2001, Walker et al., 2004]. HMMs have been described for example in [Rabiner, 1989], see these references for descriptions of Markov processes.

Basically a HMM models a time sequence as a piecewise stationary process, exactly the way we model speech by extracting short time frames for feature calculation.

The HMM consists of a number of states tied together by transitions. The HMM consists of two stochastic processes, namely a Markov process to model the random sequence of states, and an emission probability assigned to each state modeling the features at that time step. I.e. each state of the HMM corresponds to a stochastic feature vector drawn from the corresponding emission probability density function. The Markov process is governed by the transition probabilities denoted a_{ij} in figure 5.2 denoting the probability of going from state i to state j. Based on the nature of a phoneme, ASR-systems find it suitable to use three

[1]The symbols for phonemes used here are correspond to the ones used in the CMU-dictionary: http://www.speech.cs.cmu.edu/cgi-bin/cmudict

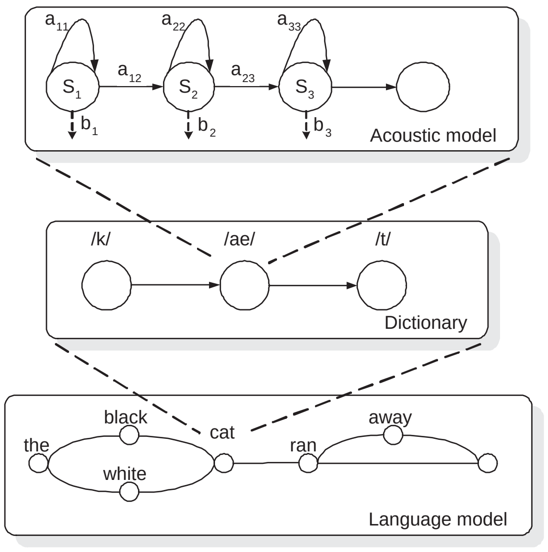


Figure 5.2: Illustration of the relationship between the recognition models. Phoneme-HMMs are concatenated into words, building sentences using the language model

state 'left-to-right' HMMs to model a single context dependent phoneme, so transitions to earlier states are excluded.

The emission probability of an observation (i.e. a feature vector) is either discrete or continuous.

- **Discrete distributions** return discrete values from an alphabet $\mathcal{A}$. Each symbol then has a probability attached depending on the HMM state. This approach requires that feature vectors are clustered to assign them to the alphabet $\mathcal{A}$.
- **Continuous distributions** are modeled by using GMM for each HMM state, yielding continuously valued outputs. Using a large number of mixture components any continuous distribution can be estimated.

Continuous models have been superior in LVASR applications, at the cost of larger computational load in decoding and training.

To use the HMMs for acoustical modeling each phoneme must have an associated HMM trained[2] on a very large corpus of annotated data, to optimize the likelihood of the models. A broadcast news transcription system may require as much as 150 hours of meticulously transcribed training data [Gauvain et al., 2002].

Using the HMMs to decode a sequence of feature vectors is done by evaluating each of the phoneme-HMMs on the feature sequence to obtain the model which yields the highest

[2]Training HMMs is a non-trivial task but will not be covered in this context. On the training algorithm see [Rabiner, 1989]

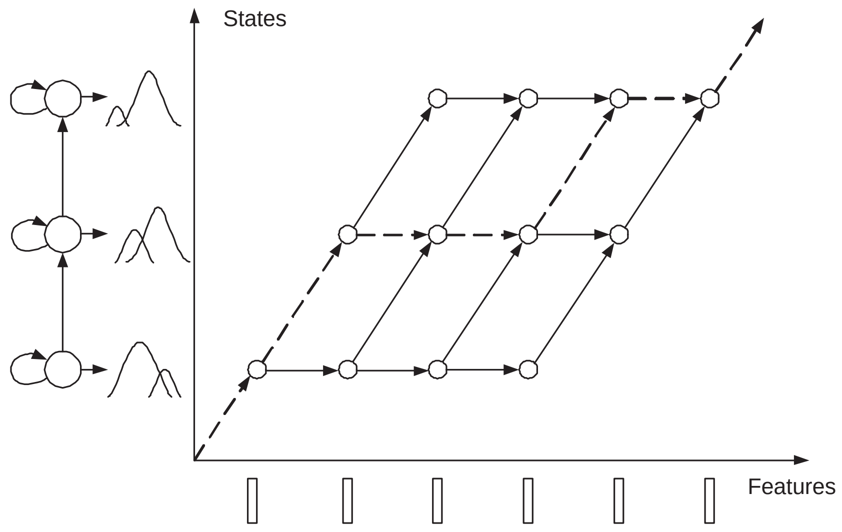


Figure 5.3: Phoneme scoring using an HMM. The HMM models the feature sequence based on the state of the markov model. Each state has a GMM attached. Evaluating the probability of a phoneme therefore is a sequence of state changes assigning each feature to a state.

probability. The evaluation of a HMM is illustrated in figure 5.3.

Language Model

The matching of phonemes must be constrained in some way, for instance 'how to recognize speech' may sound like 'how to wreck a nice beach'. Therefore the phoneme decoding is aided by a language model (LM).

The LM defines the prior probability of a sequence of words. The LM-probability of a sequence of words $(w_1, w_2, ..., w_n)$ is given by:

$$\begin{aligned} P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)...P(w_n|w_1, ..., w_{n-1}) \\ = \prod_{i=1}^{n} P(w_i|w_1, ..., w_{i-1}) \end{aligned} \tag{5.1}$$

In the expression $P(w_i|w_1, ..., w_{i-1})$, the sequence $(w_1, ..., w_{i-1})$ is the word history for w_i. In practice it is not possible to obtain reliable probability estimates given arbitrarily long histories since that would require enormous amounts of training data. Instead, the word sequence is estimated using uni-, bi-, or trigram grammars. These are defined respectively as follows:

$$\begin{aligned} P(w) &= \text{probability of word } w \\ P(w_j|w_i) &= \text{probability of } w_j \text{ given a one word history } w_i \\ P(w_k|w_i, w_j) &= \text{probability of } w_k \text{ given a two word history } w_i, w_j \end{aligned}$$

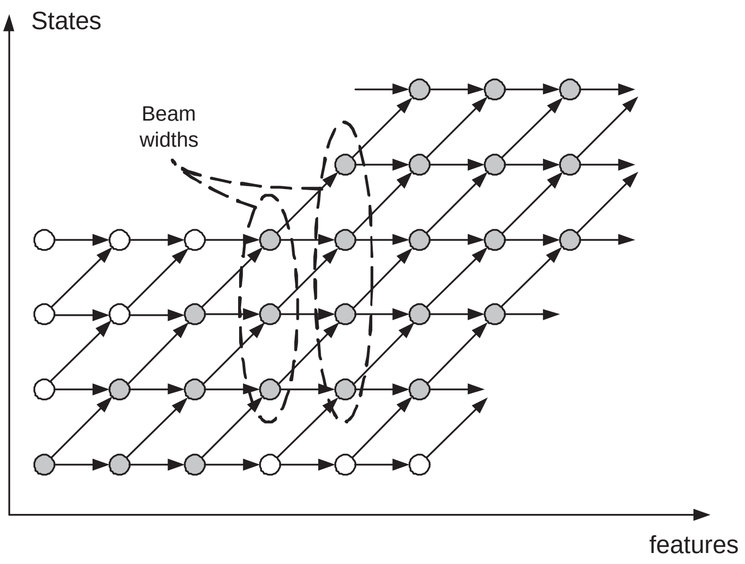


Figure 5.4: Illustration of decoding using the search graph. In general each point in the search graph has a probability assigned using Viterbi decoding based on the acoustic scoring and language model. The complexity of the Viterbi decoding is commonly reduced by applying beam pruning as illustrated, so that only the most probable paths are considered.

Other higher-order n-gram grammars can be defined similarly.

Trigram models are the most frequently used among contemporary systems, but bi- or four-gram models are also seen. The n-gram models have been the most popular because they can be trained almost unassisted, i.e. without any knowledge of language constructs, given enough appropriate training data.

5.2.3 Decoder

The acoustic model, dictionary, and language model, can be tied together to form a very large trellis or search graph, where each transition has an associated transition probability, based on the acoustic and language models.

The objective of the decoding is thus to map the sequence of features onto a path through the graph, i.e. find the optimal sequence of states and thereby aligning the input onto phonemes and words.

Searching the trellis is most commonly done by Viterbi decoding. The Viterbi algorithm processes all states completely at time t before moving on to time $t+1$. The search graph for the search is illustrated in figure 5.4. One dimension represents the states in the network, and the other represents the feature sequence. The Viterbi algorithm assigns a score to each point in this 2-D space. This score represents the probability of the best path leading up to that state. That is, given a time t and state s, the value at (t, s) represents the probability corresponding to the best state sequence leading from the initial state at time $t = 0$ to state s at time t.

Observing all paths through the search graph, though, is not feasible, as it would lead to a practically infinite search space. Therefore paths at time t are pruned out based on some heuristic. This procedure is called beam pruning, and the resulting algorithm is thus called Viterbi beam search.

5.3 The SPHINX-4 System

The introduction presented some requirements for the ASR-system. The systems mentioned in the introduction are all capable of speaker independent large vocabulary recognition.

The choice to use SPHINX-4 was made because it is provided with a number of pre-trained acoustic and language models. One of these models is trained on the HUB4 corpus which is specially tailored to evaluate ASR performance on broadcast news. The Sonic system is also provided with a number of models but the only LVASR acoustic model provided was trained on the Wall Street Journal Corpus, containing read speech in noiseless environments. The SPHINX suite was thus the only system providing acoustic models for the versatile environments seen in broadcast news.

The SPHINX systems are open-source systems that started out at Carnegie Mellon University. Different versions have been developed to investigate various applications and approaches to recognition. SPHINX-2 is for fast, almost real-time recognition, with a limited recognition accuracy. SPHINX-3 and SPHINX-4 are capable of large vocabulary speech recognition at comparable recognition rates and speed. SPHINX-3 is written in C, while SPHINX-4 is written entirely in JAVA.

Another plus of the SPHINX-4 system is that it was developed with focus on modularity, which makes it easily configurable and versatile. This section will review the SPHINX-4 system based on [Walker et al., 2004].

The system architecture is shown schematically in figure 5.5, and generally has the same structure as described above. The description below refers to the designations in the figure.

5.3.1 Frontend

The `FrontEnd` of SPHINX-4 is a pipeline performing the following tasks:

Speech extraction:
: This part does silence removal based on the energy values of the signal.

Cepstral coefficient extraction:
: The cepstral analysis is done as described in section 2.6, yielding 13 MFCCs

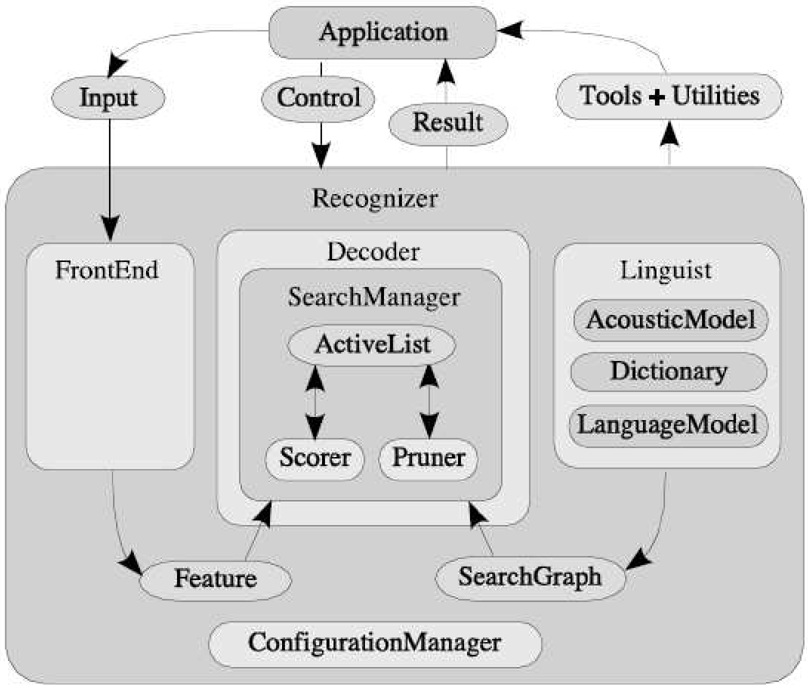


Figure 5.5: SPHINX-4 system overview. Figure from Walker et al. [2004].

including log energy as feature vectors. In addition ΔMFCCs and ΔΔMFCCs are used to obtain a 39 dimensional feature vector for each frame.

Normalization:
: To make the speech recognition more robust to different channel conditions Cepstral Mean Normalization (CMN) is applied. Running in batch-mode the mean is calculated over the entire audio stream passed on to the frontend.

5.3.2 Linguist

The SPHINX-4 organization includes the module `Linguist`, that builds a search graph based on the acoustic and language models as well as the dictionary.

Acoustic model

The `AcousticModel` used in SPHINX-4 are made using HMMs that can have an arbitrary number of states, depending on the implementation. The SPHINX-suite includes the program SphinxTrainer, that is used to train an acoustic model.

Training acoustic models for broadcast news, because of the broad range of speakers and environments, must include very large amounts of data, as mentioned above. Collecting representative data and transcribing them is a major task, and was not tractable for a project like ours. This tedious task is tackled as SPHINX-4 provide pre-trained models. Especially one trained on the large HUB4 corpus containing broadcast news. To reduce the number of tri-phones to be considered in the decoding, clustering is employed. In SPHINX-4 the set of the tri-phone HMM and the matching output probability distribution models is called senones. The acoustic model used for our system uses 6000 senones with continuous density three-state HMMs, using GMMs with 8 Gaussian components per state.

LanguageModel

The `LanguageModel` is trained using text corpora preferably from the same source as the context the system will work in. Typically this means using as many as 100 million words to train on. These may be obtained from text sources, such as news papers. The LM used in this setup was trained on the HUB4 corpus using a dictionary of $\sim 64,000$ words.

The HUB4 LM uses tri-grams. The SPHINX-system has a number of classes to handle the models, to optimize the speed and memory requirements. The optimal class for large vocabulary tri-gram models is called `LargeTrigramModel`. This model is especially optimized to handle the memory management, as the HUB4 LM takes up more than 100 MB.

Building the search graph, is as mentioned before, a combination of the acoustic and language models. This combination is handled by the `Linguist` class. In a simple system the linguist can build a static graph containing all possible combinations. This is simply not possible for LVASR, therefore the graph must be built dynamically when the decoding of the speech features advances. The `LexTreeLinguist` does just this.

The lex(ical) tree representation is a compact method to represent the search graph. The implications of using the lex tree representation are quite elaborate, for interested readers cf. to [Ravishankar, 1996, Chap. 4]. The use of context-dependent tri-phones and the lex tree leads to quite intricate problems but these will not be covered here.

5.3.3 Decoder

The search graph constructed by the `Linguist` is passed on to the `Decoder` that handles the actual decoding. As mentioned above the most tedious task lies in this search. Search is implemented using `SearchManager`, which in this system is implemented by the `WordPruningBreadthFirstSearchManager`. The search algorithm is done by assigning states to features, as described in section 5.2.3, discarding states that are least probable, thereby reducing the search space using the `Pruner`.

5.3.4 SPHINX-4 Parameter Setup

The practical setup of SPHINX-4 is done using the `ConfigurationManager` which is actually an XML-file. The configuration used for our system is listed in appendix D. There are a number of adjustable parameters, but as we did not have very much transcribed material at our disposal, an intensive test of the system was not possible. Instead we have relied on parameters found on the SPHINX-homepage [Sphinx-4, 2004] for typical settings for LVASR.

Parameters for the frontend-processing were more or less fixed due to the pre-trained acoustic model. The main parameters to tune in the system are therefore the ones used in the decoding. As mentioned before search is done using Viterbi beam search. Adjusting the parameters that control the beamwidths used in the search are the primary way to adjust both the speed and accuracy of the recognition.

The beamwidth parameter settings are listed below. As we know nothing of word duration, new word generation is also governed by probabilities, and the probability of word changes is set by the parameter `wordInsertionProbability`. The last important parameter is the `languageWeight`, which sets the weight of the language model compared to the phone scores. Setting this to a higher value means that word sequence probability is emphasized over the actual phone scores.

- `absoluteBeamWidth=20000`
- `relativeBeamWidth=1E-80`
- `absoluteWordBeamWidth=20`
- `relativeWordBeamWidth=1E-30`
- `wordInsertionProbability=0.01`
- `languageWeight=6.0`

SPHINX-4 also includes a number of monitors to retrieve performance information, such as runtime and word recognition. The setup of these monitors is also seen in the XML-file

5.4 Recognition Evaluation Measures

When evaluating ASR-systems the output sentence (hypothesis) from the ASR-system must be aligned with the true transcription (reference). When this is done, three error types can occur:

Substitutions:
: Cover words that are recognized wrongly. These include words that are in plural instead of singular.

Insertions:
: Extra words in the hypothesis that are not present in the reference.

Deletions:
: Are words present in the reference but not in the hypothesis.

Words	Matches	Errors	Subs	Ins	Del	WER(%)	WA(%)
1301	932	387	227	18	142	29.75	71.64

Table 5.1: Speech recognition performance using the audio track from an ABC Night Line TV news show. The audio track is 7 minutes and 40 seconds long.

Quantification of the errors is done using the two measures, word error rate (WER) and word accuracy (WA), defined as:

$$\text{WER} = \frac{\textit{total word errors}}{\textit{total words}}$$

$$\text{WA} = \frac{\textit{total words} - \textit{substitutions} - \textit{deletions}}{\textit{total words}}$$

where

$$\textit{total word errors}=\textit{substitutions}+\textit{insertions}+\textit{deletions}$$

and *total words* is the number of words in the reference.

5.5 Example

The performance of SPHINX-4 was tested using the audio track from an ABC Night Line TV news show. A transcription of the TV show was obtained from ABC's homepage. The audio track has a length of 7 minutes and 40 seconds. SPHINX-4 was setup using the parameters listed above.

Table 5.1 lists the performance of SPHINX-4. A total WER of 29.75% and a WA of 71.64% was obtained.

Figure 5.6 contains a sample output from SPHINX-4, where the reference sentence `REF` and the hypothesis `HYP` are listed. Below the `ALIGN_REF` lists the reference sentence where the substitutions are written with capital letters. Also the `ALIGN_HYP` are listed where again substitutions are written with capital letters and deletions are marked with stars. `ALIGN_REF` and `ALIGN_HYP` are aligned, thus the difference between the reference and the hypothesis sentences is easy to see.

5.6 Discussion

The use of the SPHINX4 system was obvious for two reasons. The setup was easy because of the modular design and easy configuration.

The other and main reason to use the system was the pre-trained acoustic and language models trained with the HUB4 corpus consisting of broadcast news audio. This was a huge advantage for us as it made it possible to get reasonable results for the broadcast news data used in our evaluations. The pretrained models on the other hand constrains the tuning

```
REF:       since nineteen ninety eight more than two hundred and sixty
people have died in terrorist attacks on u. s. diplomatic facilities

HYP:       since nineteen ninety eight with two hundred sixty
people have died in terrorist attacks on u. s. diplomatic tussle


ALIGN_REF: since nineteen ninety eight MORE THAN two hundred AND sixty
ALIGN_HYP: since nineteen ninety eight WITH **** two hundred *** sixty

people have died in terrorist attacks on u. s. diplomatic FACILITIES
people have died in terrorist attacks on u. s. diplomatic TUSSLE


   Accuracy: 80.952%    Errors: 4  (Sub: 2  Ins: 0  Del: 2)
   Words: 21   Matches: 17    WER: 19.048%
```

Figure 5.6: Output sample from SPHINX-4. The output contains the reference and hypothesis sentences. Also the aligned reference and hypothesis are listed.

of parameters of the system but it was simply not feasible to train our own models given the limited amount of data collected in our databases. Furthermore it was actually not possible to do a thorough test of the speech recognition performance because the amount of transcribed material was limited.

Chapter 6

Full System Example

In this chapter the audio classification, speaker change detection, and speech recognition will be combined to a full system. The system performance will be illustrated through an example.

6.1 System Setup

Figure 6.1 shows the system setup. First the features are extracted on a 20 ms basis as described in chapter 2 and propagated to the audio classification and speaker change detection parts.

Audio classification is performed to find the speech segments of the audio stream for further processing while music segments are not processed further in this system. For this part the linear discriminant described in chapter 3 is used. The linear discriminant is trained on the audio-database containing 2250 seconds of speech and 2250 seconds of music.

The extracted speech segments are further processed in the speaker change detection algorithm, where the speech is segmented into speaker segments. For this we use the change detection algorithm using the VQD measure with 32 clusters as described in chapter 4.

The generated speaker segments are finally passed on to the SPHINX-4 speech recognizer to make transcriptions of the individual segments. SPHINX-4 is setup as described in chapter 5.

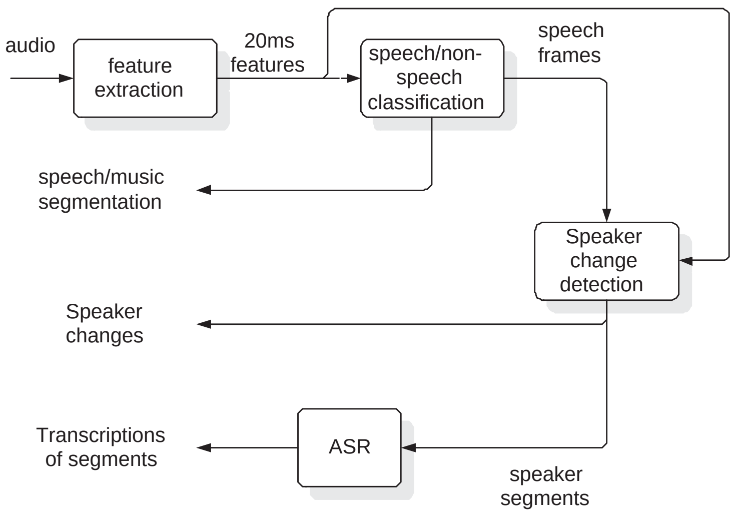


Figure 6.1: The setup of the full system.

6.2 Example

The audio stream used for this example is the podcast edition of the CNN news update from December 5th, 2005 at 7AM. The audio stream is 118 seconds long and contains an 1.9 seconds long intro jingle and an 2.8 seconds long outtro jingle. The stream contains 9 speaker segments, that is 8 speaker changes. 5 different speakers are present. Table 6.1 shows the start and end time and a description for each segment.

6.2.1 Audio Classification

The first part of the system is the extraction of speech segments. Figure 6.2 shows how the system segments the audio stream into the classes speech and music. The system segments $0-2$ s as music $2-115$ s as speech and $115-118$ s as music. Thus, the two jingle segments are detected and removed, and no other speech segments are detected, which is in accordance with the true segmentation.

6.2.2 Speaker Change Detection

The next part of the system is the speaker change detection. The upper part of figure 6.3 shows the $\mathrm{VQD_n}$ measure for the speech part of the audio stream. Potential speaker change-points are marked with circles. The dotted line indicate the threshold $\mathrm{th_{cd}}$. The lower part of the figure shows the $\mathrm{VQD_{mean}}$ for the found change-points. Again, the dotted line indicate the threshold $\mathrm{th_{fac}}$. The accepted change-points are marked with circles and the rejected change-points are marked with crosses.

No.	Time (s)	Description	Gender	Environment
-	0.0 - 1.9	jingle	-	-
1	1.9 - 20.7	anchor	male	studio
2	20.7 - 32.5	speaker 1	male	press conference
3	32.5 - 48.4	anchor	male	studio
4	48.4 - 62.8	reporter 1	male	studio
5	62.8 - 67.6	speaker 2	male	press conference
6	67.6 - 83.5	reporter 1	male	studio
7	83.5 - 91.4	anchor	male	studio
8	91.4 - 102.9	reporter 2	male	telephone
9	102.9 - 115.2	anchor	male	studio
-	115.2 - 118.0	jingle	-	-

Table 6.1: Times and descriptions for the segments of the audio stream used for this example.

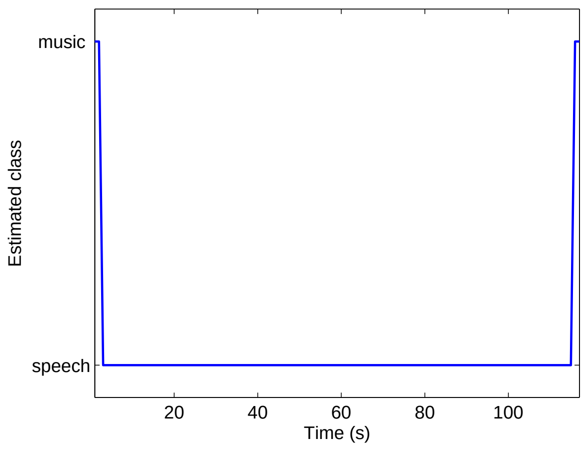


Figure 6.2: This figure shows how the audio is segmented into speech and music.

All true speaker change-points are found by our system and no false alarms occur. The change-points are found at 20.7, 32.7, 48.0, 62.2, 67.8, 83.6, 91.5, and 103.1 seconds. This gives an average mismatch of 0.225 seconds.

6.2.3 Speech Recognition

The final part of the system is the speech recognition. Two approaches have been investigated for the speech recognition. In the first approach the full speech segment is processed in SPHINX-4. The second approach divides the speech into speaker segments and process each segment individually in SPHINX-4. The segmentation is based on the change-points found by the speaker change detection algorithm.

Table 6.2 shows the word accuracy (WA) and word error rate (WER) for each of the speaker

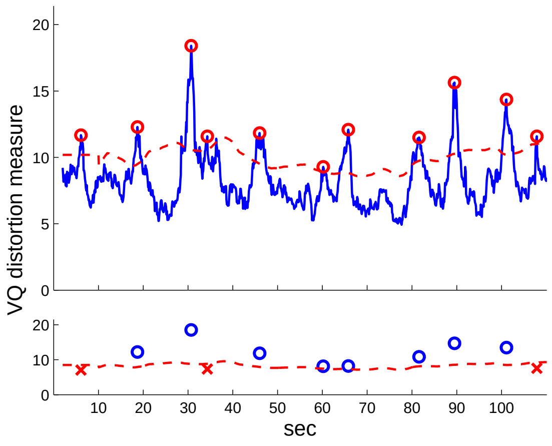


Figure 6.3: The upper part of the figure shows the VQD_n measure for the speech part of the audio stream. The threshold th_{cd} is marked with dotted line, and the potential change-points are marked with circles. The lower part of the figure shows the VQD_{mean} measure for the potential change points. Again, the dotted line indicates the threshold th_{fac} and the accepted change-points are marked with circles and the rejected change-points are marked with crosses. In this case all true change-points are found, and no false alarms occur.

segments in the audio stream. The table shows that a small improvement is obtained by segmenting the audio stream into speaker segments.

As seen in the table an overall WA of 75.1% and WER of 28.3% is obtained. For the anchor speaker the WA is 80.2% and WER is 23.6% and for reporter 1 WA is 82.8% and WER 19.4%. The performance of the speech recognition is seen to degrade for segment 2 and segment 8. Segment 2 contains speech from a press conference, where background noise is present and in addition figure 6.4 shows that the bandwidth for this segment is limited. Segment 8 is a telephone report from Israel. Listening to this segment clearly reveals that signal has been transmitted through a telephone line, where compression has been applied. Thus, this show that clear speech with full bandwidth gives the best recognition rate.

Appendix E lists the true transcriptions and the output from the ASR for each of the speaker segments in the case where no segmentation was done.

The runtime of the speech recognition was 1091 seconds on an 113 seconds long audio stream. This means that the speech recognition can be done in a little less than 10 times realtime.

6.3 Discussion

We have showed that the system is capable of producing an useable output for indexing and retrieval tasks.

		no speaker segmentation		speaker segmentation	
No.	Description	WA (%)	WER (%)	WA (%)	WER (%)
1	anchor speaker male	84.8	18.2	84.8	18.2
2	press conference 1 male	48.2	51.7	48.3	55.2
3	anchor speaker male	72.0	34.0	76.0	30.0
4	reporter 1 male	90.4	9.5	88.1	11.9
5	press conference 2 male	78.5	28.5	78.5	35.7
6	reporter 1 male	82.3	21.5	78.4	25.5
7	anchor speaker male	78.5	21.4	82.1	17.9
8	reporter 2 male	53.8	46.1	51.3	58.7
9	anchor speaker male	60.5	39.4	76.3	28.9
	total	73.6	28.5	75.1	28.3

Table 6.2: Word accuracy (WA) and word error rate (WER) for the 9 speech segments. Both the results obtained with and without segmenting the audio stream into speaker segments before processing in the speech recognizer are shown.

The system performs best by dividing the audio into speaker segments. The advantage of doing this segmentation most likely comes from the use of cepstral mean normalization (CMN). This method clearly works best if it is applied on homogenous data, i.e. speech from one speaker in one environment. The disadvantage of doing speaker segmentation is the potential mismatch between true change-points and the change-points found by the speaker change detection algorithm. If a change-point is not correctly estimated the first/last words could be destroyed in the segmentation. However this seems not to be a problem in this example, as the average mismatch is relatively small.

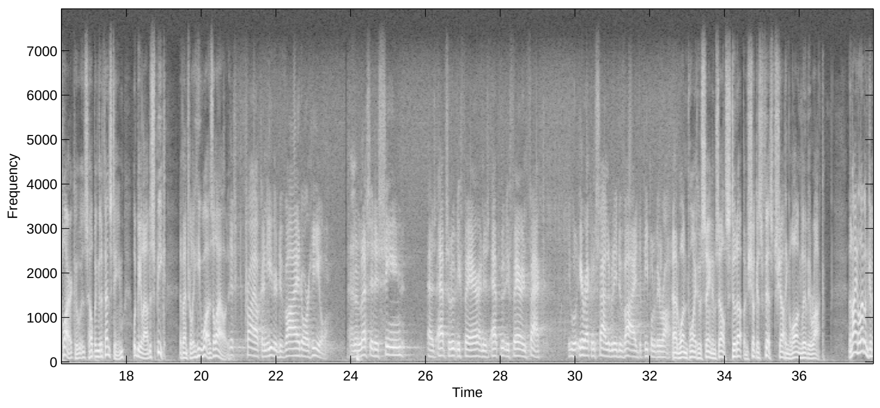


Figure 6.4: Spectrogram of the speech where it can be seen that the bandwidth in the segment from $20.7 - 32.5s$ is limited.

Chapter 7

Conclusion

This project has investigated and implemented methods for an audio indexing using segmentation and transcription system for audio retrieval. The system includes audio classification, speaker change detection, and speech recognition. The three parts of the system were implemented and evaluated separately. Finally the system was combined and evaluated on an example.

A number of features were investigated, and through integration over 1 second windows using means and variance, a total number of 60 features were proposed. Using these 60 features several classifiers where investigated. The classifiers all showed very good performance, giving a test misclassification rate of 2.6% to 3.4% using a feature set of 60 proposed features. Two feature selection schemes, based on NN and the linear discriminant respectively, where developed for finding an optimal set of features. The NN pruning scheme showed that the number of features could be reduced to 10 without increasing the test misclassification rate. The exact set of features to use was not clearly evident as several feature combinations where able to get equal performance. The feature selection scheme based on the linear discriminant classifier also showed that the number of features could be decreased to about 14 without decrease in test performance. The backward elimination and forward selection showed almost same performance. The model finally chosen for audio classification, was the computationally simple linear discriminant with a 14-dimensional feature.

We have developed a speaker change detection algorithm based on the vector quantization distortion measure. The change detection algorithm works in two steps. The first step finds potential speaker change-points. The second step is a false alarm compensation step, that uses longer segments to build more reliable models, and thereby either accept or reject potential change-points. The optimal parameters for the speaker change detection was found, and the performance using VQD was compared with the two other frequently used metrics KL2 and DSD. This comparison showed that the VQD metric was superior to the other two metrics. The best performance was observed using VQD with 56 clusters, where an F-measure of 0.854 was obtained. The improved performance using VQD, compared to KL2 and DSD, comes with a cost in computational runtime. We showed that a relative improvement of 59.7% in precision with a relative loss of 7.2% in recall is obtained with our false alarm compensation scheme. The generalizability of our proposed algorithm was investigated, and showed that the choice of the thresholds based on one data set generalized reasonably well to other different data sets from different stations.

SPHINX-4 was selected as the speech recognition system for the transcription part of our system. SPHINX-4 is an open source, speaker independent system capable of doing large vocabulary speech recognition. The system was adapted for broadcast news transcription using the pre-trained acoustic and language models trained on the HUB4 corpus. SPHINX-4 was setup using the typical parameters for large vocabulary recognition. The performance was demonstrated on a 7 minutes and 40 seconds long audio track from a TV show, where a total word accuracy of 71.6% was obtained.

Finally an example of the full system was given on a CNN news show. This example showed that the system was able to detect speech segments and further segment this into speaker segments. The example showed an improved word recognition performance by segmenting the audio into speaker segments before processing in the ASR. An overall word accuracy of 75.1% was found.

7.1 Further Work

The segmentation provided by our system provides a starting point for further extraction of information.

The implemented speaker change detection takes an unsupervised approach. The algorithm finds change-points between two speakers producing speaker segments. These segments could be further processed. Clustering the speaker segments could gather the speech one speaker has uttered along the show. In this way a recurring speaker, e.g. an anchor, could be picked out. Speaker detection could be applied if speaker identities are known a priori. In broadcast news it would for instance be beneficial to make a database of the anchor speakers. This could improve the segmentation performance.

The performance of the speech recognition could be increased by speaker or environment adaption. This would require that more specific acoustic models should be trained. Methods to adapt speaker independent acoustic models to specific speakers have been proposed. Also language model adaption could be considered. If the topics of the speech is known a priori, related words and word combinations could be made more probable in the language model.

In this project the only information extracted from the speech is the transcription and speaker changes. Other information such as gender, mood, accent, age, etc. could make the audio retrieval more valuable.

Appendix A

Eusipco 2006 Paper

This appendix contains the paper: Unsupervised Speaker Change Detection for Broadcast News Segmentation, which we have submitted to the European Signal Processing Conference (EUSIPCO) 2006. The paper can also be found at:

`http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=4416`

UNSUPERVISED SPEAKER CHANGE DETECTION FOR BROADCAST NEWS SEGMENTATION

Kasper Jørgensen, Lasse Mølgaard, and Lars Kai Hansen

Informatics and Mathematical Modelling, Technical University of Denmark
Richard Petersens Plads, Building 321, DK-2800 Kongens Lyngby, Denmark
phone: +(45) 4525 3889, fax: +(45) 4587 2599, email: s001498,s001514,lkh@imm.dtu.dk,
web: http://isp.imm.dtu.dk

ABSTRACT

This paper presents a speaker change detection system for news broadcast segmentation based on a vector quantization (VQ) approach. The system does not make any assumption about the number of speakers or speaker identity. The system uses mel frequency cepstral coefficients and change detection is done using the VQ distortion measure and is evaluated against two other statistics, namely the symmetric Kullback-Leibler (KL2) distance and the so-called 'divergence shape distance'. First level alarms are further tested using the VQ distortion. We find that the false alarm rate can be reduced without significant losses in the detection of correct changes. We furthermore evaluate the generalizability of the approach by testing the complete system on an independent set of broadcasts, including a channel not present in the training set.

1. INTRODUCTION

The increasing amount of audio data available via the Internet emphasizes the need for automatic sound indexing. Broadcast news and other podcasts often include multiple speakers in widely different environments. Efficient indexing of such audio data will have many applications in search and information retrieval. Segmentation of sound streams is a significant challenge including segmentation of sequences of music and different speakers. Locating parts that contain the same speaker in the same environment can indicate story boundaries and may be used to improve automatic speech recognition performance. Indexing based on speaker recognition is a possibility but is hampered by the prevalence of unknown speakers, thus we have chosen to investigate unsupervised methods in this work in line with other recent systems, see e.g., [1]. Here we are interested in systems that are not too specialized to a given channel, hence, in both system design and in the evaluation procedure we will focus on the issue of robustness. In particular we show that a system can be tuned to a set of channels and not only generalize to other broadcasts from these channels, but also to a channel not present in the training set.

Speaker change detection approaches can roughly be divided into three classes: energy-based, metric-based and model-based methods. Energy-based methods rely on thresholds on the audio signal energy, placing changes at 'silence' events. In news broadcast the audio production can be quite aggressive with only little if any silence between speakers, making this approach less attractive.

Metric based methods basically measure the difference between two consecutive frames that are shifted along the audio signal. A number of distance measures have been investigated such as the symmetric Kullback-Leibler distance [2]. Parametric models corrected for finite samples using the Bayesian Information Criterion (BIC) are also widely used. Huang and Hansen [3] argued that BIC-based segmentation works well for longer segments, while BIC approach with a preprocessing step that uses a T^2-statistic to identify potential changes, was superior for short segments.

Nakagawa and Mori [4] compare different methods for change detection, including BIC, Generalized Likelihood Ratio, and a vector quantization (VQ) based distortion measure. The comparison indicates that the VQ method is superior to the other methods.

A simplification of the Kullback-Leibler distance, the so-called divergence shape distance (DSD), was presented in [1] for a real-time implementation. The system includes a method for removing false positives using "lightweight" GMM speaker models.

Model-based methods are based on recognizing specific known audio objects, e.g., speakers, and classify the audio stream accordingly. The model-based approach has been combined with the metric-based to obtain hybrid-methods that do not need prior data [5][6].

Our basic sound representation is the mel-weighted cepstral coefficients (MFCC), they have shown useful in a wide variety of audio application including speech recognition, speaker recognition [7] and music modelling, see e.g., [8].

Since we are interested in segmenting news with an unknown group of speakers we limit our investigation to metric based methods. To improve the performance we invoke a false alarm compensation step at relative low additional cost.

2. DISTANCE MEASURES

Metric based change detection is done by calculating a distance between two successive windows. The distance indicates the similarity between the two windows. Below we present three different distance measures that have been considered in this context.

2.1 Vector Quantization Distortion

The VQ approach is based on the generalized distance between two feature vectors sequences designated $\mathrm{S^A}$ and $\mathrm{S^B}$.

The VQ-distortion measure VQD between $\mathrm{S^B}$ and the codebook $\mathrm{C^A}$, created by clustering of the features in $\mathrm{S^A}$, is defined as:

$$\mathrm{VQD}(\mathrm{C^A},\mathrm{S^B}) = \frac{1}{\mathrm{T}}\sum_{\mathrm{t}=1}^{\mathrm{T}} \underset{1\le \mathrm{k}\le \mathrm{K}}{\operatorname{arg\,min}} \left\{ \mathrm{d}\left(\mathrm{C^A_k},\mathrm{S^B_t}\right)\right\},$$

where C^A_k denotes the k-th code-vector in C^A, $1 \leq k \leq K$. S^B_t denotes the t-th feature vector in the sequence S^B, $1 \leq t \leq T$, and d is the *Euclidean* distance function, see e.g., [1].

The codebook C^A is created by clustering the sequence of feature vectors S^A into K clusters, thus each cluster-center represents a code-vector.

2.2 Kullback-Leibler Distance

The symmetric Kullback-Leibler distance (KL2) has been used in speaker identification systems and applied to speaker change detection [9]. The symmetric Kullback-Leibler distance between two audio segments represented by their feature vector sequences S^A and S^B is defined as:

$$\mathrm{KL2}(S^A, S^B) = \int_{\boldsymbol{x}} [p_A(\boldsymbol{x}) - p_B(\boldsymbol{x})] \log \frac{p_A(\boldsymbol{x})}{p_B(\boldsymbol{x})} d\boldsymbol{x} \qquad (1)$$

Assuming that the feature sequences S^A and S^B are n-variate Gaussian distributed, $p_A \sim \mathcal{N}(\boldsymbol{\mu}_A, \boldsymbol{\Sigma}_A)$, $p_B \sim \mathcal{N}(\boldsymbol{\mu}_B, \boldsymbol{\Sigma}_B)$, i.e.

$$p(\boldsymbol{x}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{ -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) \right\} \qquad (2)$$

Combining equation (1) and (2) gives:

$$\begin{aligned} \mathrm{KL2}(S^A, S^B) &= \frac{1}{2}\mathrm{Tr}\left[(\boldsymbol{\Sigma}_A - \boldsymbol{\Sigma}_B)(\boldsymbol{\Sigma}_B^{-1} - \boldsymbol{\Sigma}_A^{-1})\right] \\ &\quad + \frac{1}{2}\mathrm{Tr}\Big[(\boldsymbol{\Sigma}_A^{-1} + \boldsymbol{\Sigma}_B^{-1})(\boldsymbol{\mu}_A - \boldsymbol{\mu}_B) \\ &\quad (\boldsymbol{\mu}_A - \boldsymbol{\mu}_B)^\top\Big] \end{aligned}$$

2.3 Divergence Shape Distance

The KL2 distance presented above is composed of two terms. The last term depends on the means of the features which can vary much depending on the environment [1]. Using only the first term should remove this dependency, so that only the difference between covariance contribute. This function is called the divergence shape distance (DSD).

$$\mathrm{DSD}(S^A, S^B) = \frac{1}{2}\mathrm{Tr}\left[(\boldsymbol{\Sigma}_A - \boldsymbol{\Sigma}_B)(\boldsymbol{\Sigma}_B^{-1} - \boldsymbol{\Sigma}_A^{-1})\right]$$

In all of the three presented distance measures a greater value means a greater difference in the two distributions.

3. SPEAKER CHANGE DETECTION

Based upon the distance metric the change detection algorithm determines whether or not a speaker change occurred.

Our algorithm works in two steps. The first step is the change-point detection part where candidate change-points are found. The second step is the false alarm compensation step.

3.1 Front-End Processing

MFCCs are chosen as the features for this work. The calculation of these features is preceded by transforming the audio streams to a common sampling and bitrate.

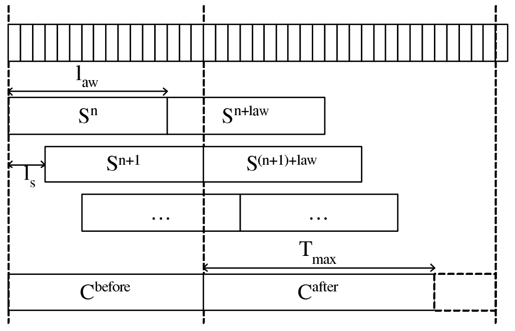


Figure 1: Illustration of windows used in the metric calculation. Speaker change-points are indicated with vertical dashed lines. The figure assumes that a change is found at time t_{n+1}, and false alarm compensation windows are shown at the bottom

3.2 Distance Metric Calculation

The audio is divided into analysis windows of length l_{aw} and with a shift of length l_s, see figure 1. Let S^n denote the sequence of feature vectors extracted from the analysis window with endtime t_n. Then, S^n and $S^{n+l_{aw}}$ are two succeeding and non-overlapping analysis windows.

For each feature vector sequence S^n a codebook C^n is created by clustering the vector sequence into K clusters using the *k-means* clustering algorithm. Convergence of the *k-means* algorithm is sped up by exploiting the overlap of the analysis windows, which means that most samples are reused in subsequent analysis windows. The code-vectors of C^n are therefore computed using the code-vectors from C^{n-l_s} as initial cluster centers. This makes the *k-means* algorithm converge faster and minimizes the distance between two succeeding codebooks, resulting in less fluctuating distortion measures.

The conventional VQ-algorithm computes the distortion measure between two feature vector sequences S^A and S^B by computing $\mathrm{VQD}(C^A, S^B)$. By using the code-vectors of C^B instead of the whole sequence S^B, better results are obtained. Thus, we use $\mathrm{VQD}_n = \mathrm{VQD}(C^{S^n}, C^{S^{n+law}})$ as the VQ-distortion measure at time t_n.

The $\mathrm{KL2}_n$ and DSD_n at time t_n are given by $\mathrm{KL2}_n = \mathrm{KL2}(S^n, S^{n+l_{sw}})$ and $\mathrm{DSD}_n = \mathrm{DSD}(S^n, S^{n+l_{sw}})$

3.3 Change-Point Detection

The basic change-point detection evaluates the calculated distance metric M_n at every time step time (t_n). A change-point is found if M_n is larger than a threshold th_{cd} and M_n is the local peak within T_i seconds. The intention of this baseline approach is to detect as many true change-points as possible. The false alarms that occurs should then be rejected by our false alarm compensation described below.

3.4 False Alarm Compensation

When running the speaker change-point detection algorithm it is necessary to keep the analysis window relatively short in order to be able to detect short speaker turns. The short segments may lack data to make fully reliable segment models, which consequently may cause false alarms.

The baseline approach yields a number of potential change-points, dividing the audio stream into speaker segments. These speaker segments can then be used to make more accurate models between the potential change-points. Comparing these models can then accept or reject the potential change-point.

The false alarm compensation algorithm simply works by making two speaker VQ-codebooks, for the speaker segment before the change-point C^{before} and another after the change-point C^{after}.

The two VQ-distortion measures $VQD(C^{before}, C^{after})$ and $VQD(C^{after}, C^{before})$ are computed and the mean VQD_{mean} of these two measures is found. The change-point is then accepted if the measure is larger than the threshold th_{fac} and rejected if it is below. We found that using the mean of the two distortion measures is more stable than using just one of the measures.

If a real speaker change is missed during the initial change-point detection, the resulting speaker model would contain data from two speakers, meaning that the speaker codebook models both speakers. To counteract this problem only the T_{max} seconds nearest the change-point is used to make the speaker codebook.

3.5 Parameter Settings

The proposed change-point detection algorithm requires some parameters to be adjusted. The two thresholds th_{cd} and th_{fac} should be set according to the desired relation between recall and precision. As in [1] we use an automatic threshold setting method. We use $M_{n,mean}$ as the mean of the distance metric in a window of $2T_{max}$ around t_n:

$$M_{n,mean} = \frac{1}{2T_{max}+1}\sum_i M_{n+i},$$

with $-T_{max}/l_s < i < T_{max}/l_s$. The thresholds at time t_n are thereby set to:

$$\begin{aligned} th_{cd,n} &= \alpha_{cd} M_{n,mean} \\ th_{fac,n} &= \alpha_{fac} M_{n,mean} \end{aligned}$$

The two amplifiers α_{cd} and α_{fac} should be set in advance.

The timing parameters l_{aw}, T_i, and T_{max} should be set according to the expected distribution of speaker turn lengths. l_s defines the resolution of the detected change-points.

3.6 Example

An example of the change-point detection algorithm is shown in figure 2. The audio clip in this example is $113s$ long and contains speaker change-points at time $t = \{14.6, 29.3, 33.7, 43.8, 63.5, 78.9\}s$ indicated by the vertical lines. The upper part of the figure shows the VQ-distortion measure VQD_n as function of time. The dotted line indicate the threshold th_{cd} and the estimated change-points found by our change-point algorithm are shown with circles. It is seen that in addition to the true speaker change-points four false false alarms occur.

The lower part of the figure shows the VQ-distortion measure VQD_{mean} for the found change-points. Again, the dotted line indicate the threshold th_{fac} and the accepted change-points are shown by circles, and the rejected are shown by crosses.

In this example all the true speaker changes are found, and false alarms are removed by the false alarm compensation step.

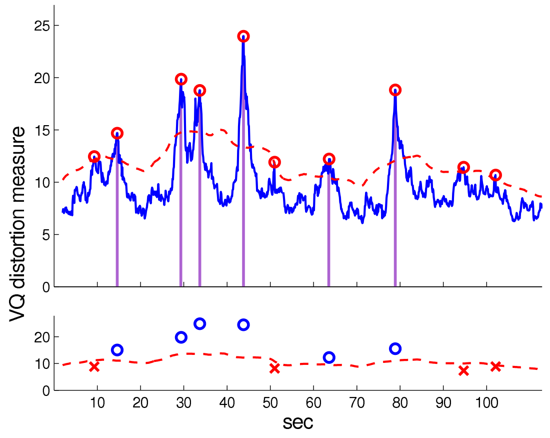


Figure 2: The upper part of the figure shows the VQ-distortion measure VQD_n for a sample file. The true speaker changes are indicated by vertical lines. The dotted line indicates the threshold th_{cd} and the estimated change-points found are shown with circles. In addition to the true speaker change-points four false change-points are found. The lower part of the figure shows the VQ-distortion VQD_{mean} for the found change-points. The threshold th_{fac} is indicated and the accepted change-points are shown by circles, and the rejected are shown by crosses.

4. EXPERIMENTS AND RESULTS

4.1 Speech Database

The speech data used was news-podcasts obtained from four different news/radio channels CNN, CBS, WNYC, and PRI.

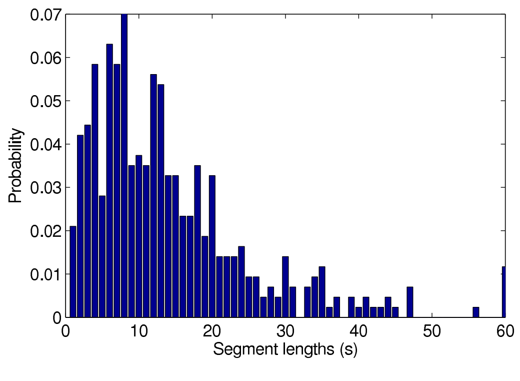


Figure 3: Histogram of the speaker segment lengths contained in the database.

The data consists of 103 min of broadcast news, which contains speech from numerous speakers, in different environments. Music has been removed as this is assumed to be done using a music/speech discriminator. The length of the segments range from $0.4s$ to $119s$ with a mean of approximately $14s$. Figure 3 shows the distribution of the segment lengths. The number of speaker changes is 388, distributed over 47 files. The data was manually labelled into different speakers. The number of segments is 435, and 75 of these have a length less than $5s$, which are segments considered relatively hard to detect [1, 3].

	Total length (min)	Avg. segment length (sec)	Speaker changes
CNN	38	17.0	134
CBS	20	9.9	121
WNYC	26	22.6	69
PRI	19	15.8	64
All	103	15.6	388

Table 1: Summary of evaluation data.

4.2 Feature Extraction

First all files have been down-sampled to 16kHz, 16bit mono channel. The MFCCs are extracted on a 20 ms Hamming filtered window. The windows overlap by 10 ms. The feature vector consists of 12 MFCCs. 'delta-MFCCs' or 'delta-delta-MFCCs' were not included because they worsened segmentation results. The features are not normalized.

4.3 Evaluation Measures

A change-point proposed by the algorithm may not be precisely aligned with the manual label. For example if the change occurs at a silence period or if speakers interrupt each other. To take this into account, a found change is counted as correct if it is within $1s$ of the manually labelled change-point, as in [3]. The *mismatch* is defined as the time between a correct found change-point point and the manually labelled one.

The evaluation measures frequently used are recall (RCL) and precision (PRC), that correspond to deletions and insertions respectively.

$$\begin{aligned} \mathrm{RCL} &= \frac{\text{no. of correctly found change-points}}{\text{no. of true change-points}} \\ \mathrm{PRC} &= \frac{\text{no. of correctly found change-points}}{\text{no. of hypothesized change-points}} \end{aligned}$$

The F-measure combines RCL and PRC into one measure,

$$\mathrm{F} = \frac{\mathrm{RCL} \times \mathrm{PRC}}{\alpha \times \mathrm{RCL} + (1-\alpha)\mathrm{PRC}}$$

with α as a weighting parameter that can be used to emphasize either of the two quantities. The results presented below use the equal weighting, with $\alpha = 0.5$.

4.4 Results

This section will present the results obtained with our speaker change detection algorithm. The length of the analysis window is set to $\mathrm{l_{aw}} = 3s$. $\mathrm{T_i}$ is set to $2s$ and $\mathrm{T_{max}}$ is set to $8s$. The analysis windows are shifted with $\mathrm{l_s} = 0.1s$.

Table 2 shows the results obtained using all the data from our database. α_{cd} and α_{fac} are set to maximize the F-measure after the false alarm compensation (FAC). The VQ-approach is evaluated using 24, 48, 56, and 64 clusters for both the change detection and in the false alarm compensation. In the KL2-FAC and DSD-FAC approaches, 56 clusters are used.

Comparing the results using the VQD measure the best performance is obtained using 56 clusters. In this case 80.1% of the true change-points are detected with a false alarm rate of 8.5 %. A relative improvement of 59,7% in precision with a relative loss of 7.2% in reduction is obtained with our false alarm compensation scheme.

By varying α_{cd} a recall-precision curve can be created. Figure 4 shows the recall-precision curve for the three metrics VQD-56, KL2, and DSD for the baseline algorithm. The curves for VQD-56 and KL2 are comparable, though VQD-56 gives better precision at lower recall. VQD-56 and KL2 is clearly better than DSD.

Figure 5 shows the recall-precision curves after the false alarm compensation. This curve is created by varying α_{cd} and keeping α_{fac} constant. Though, the baseline recall-precision curve for VQD and KL2 is very similar the VQD-FAC performs better than KL2-FAC. A reason for this could be that VQD and KL2 do not locate the same change-points and FAC then rejects more true change-point found by KL2 than found by VQD.

The change-points are found with a relatively small average mismatch of approximately $0.2s$, which is acceptable for most applications.

An investigation reveals that approximately 62% of the missed change points are due to segments that are shorter than $5s$.

Metric	F	RCL	PRC	Mismatch
VQD24	0.748	0.810	0.695	209ms
VQD24-FAC	0.829	0.740	0.943	206ms
VQD48	0.717	0.840	0.627	208ms
VQD48-FAC	0.839	0.766	0.928	206ms
VQD56	0.687	0.863	0.573	220ms
VQD56-FAC	0.854	0.801	0.915	202ms
VQD64	0.722	0.835	0.637	202ms
VQD64-FAC	0.837	0.789	0.892	215ms
KL2	0.763	0.833	0.704	212ms
KL2-FAC	0.823	0.789	0.860	212ms
DSD	0.623	0.766	0.526	308ms
DSD-FAC	0.732	0.665	0.814	288ms

Table 2: Results obtained with α_{cd} and α_{fac} adjusted to optimize the F measure after the false alarm compensation (FAC). Both the results before and after the FAC is shown.

4.5 Generalizability

To investigate the generalizability of our system, another test was set up where the database was divided into a training set and four test sets. The training set contains files randomly chosen from three of the channels, CNN, CBS, and WNYC. Four test sets were created, one for each of the channels, using the remaining files in the database.

The system was set up using the VQD measure with 56 clusters. The system parameters α_{cd} and α_{fac} were optimized for the training set and then evaluated on the test sets. Figure 6 shows the F-measure for this test. The results are compared with the system optimized for each of the specific test sets.

Generally our system performs better on the two test sets CNN and CBS compared to WNYC and PRI. This is most likely due to the fact that WNYC and PRI contain more short segments ($<3s$) than CNN and CBS. The analysis window length of $3s$ makes these segments hard to locate.

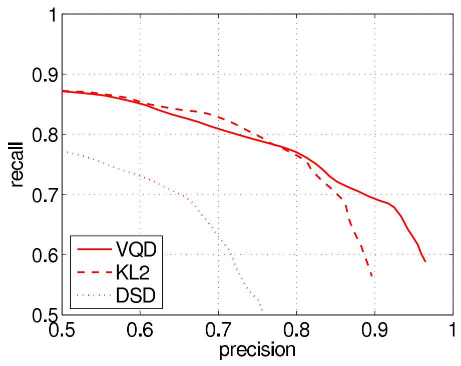


Figure 4: Recall-precision curve for baseline algorithm with the three distance metrics VQD, KL2, and DSD. The curve is created by varying α_{cd}. VQD and KL2 are superior to the DSD measure. VQD gives a better precision at lower recall rates.

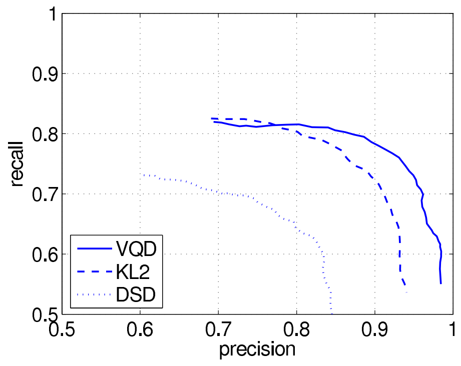


Figure 5: Recall-precision curve after the false alarm compensation with the three distance metrics VQD, KL2, and DSD. The curve is created by varying α_{cd} and keeping α_{fac} constant.

Only a minor reduction in the F-measure for all test sets is observed when using the training setting compared to the optimal settings for these test sets. Even the data from PRI that was not present in the training set show the same behavior. This demonstrates that the system is robust and lend support to the use in different media without need for further supervised tuning of parameters for new channels.

5. CONCLUSION

We have outlined an approach for robust segmentation of broadcast news. Fully implemented such a system could enable search in a broader media base than current web search engines. We have emphasized the need for an unsupervised approach because only a fraction of the speakers can be known a priori in realistic news cast. We obtained state-of-the-art performance using a vector quantization distance measure. The vector quantization approach showed better performance than systems based on the symmetric KL distance and the so-called 'divergence shape distance'. We showed that the choice of system parameters based on one data set generalized well to other independent data sets, including data from a different channel. We showed that the false alarm rate can be significantly reduced using a post-processing step on the alarms suggested by the vector quantizer.

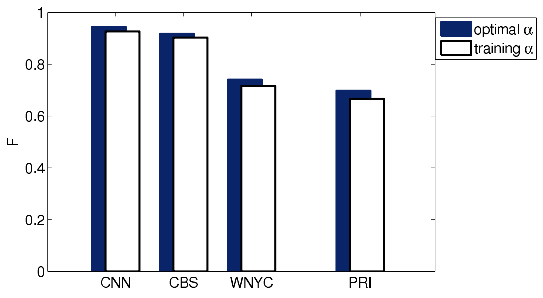


Figure 6: This figure shows the results obtained for different test sets. The system optimized for each of the tests are compared with a system optimized for a training set. The figure shows that a threshold chosen on a training set generalize reasonable well to other data sets.

Acknowledgments

This work is supported by the Danish Technical Research Council, through the framework project 'Intelligent Sound', www.intelligentsound.org (STVF No. 26-04-0092).

REFERENCES

[1] L. Lu and H. Zhang, "Unsupervised speaker segmentation and tracking in real-time audio content analysis," *Multimedia Systems*, vol. 10, no. Issue.4, pp. 332–343, 2005.

[2] M. Siegler, U. Jain, B. Raj, and R. Stern, "Automatic segmentation, classification and clustering of broadcast news audio," *DARPA Speech Recognition Workshop*, pp. 97–99, 1997.

[3] R. Huang and J. H. Hansen, "Advances in unsupervised audio segmentation for the broadcast news and ngsw corpora," *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing, 2004*, vol. 1, pp. 741–744, May 2004.

[4] S. Nakagawa and K. Mori, "Speaker change detection and speaker clustering using vq distortion measure," *Systems and Computers in Japan*, vol. 34, no. 13, pp. 25–35, 2003.

[5] T. Kemp, M. Schmidt, M. Westphal, and A. Waibel, "Strategies for automatic segmentation of audio data," *IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings*, vol. 3, pp. 1423–1426, 2000.

[6] H.-G. Kim, D. Ertelt, and T. Sikora, "Hybrid speaker-based segmentation system using model-level clustering," in *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '05)*, 2005.

[7] T. Ganchev, N. Fakotakis, and G. Kokkinakis, "Comparative evaluation of various mfcc implementations on the speaker verification task," in *10th International Conference on Speech and Computer, SPECOM 2005*, vol. 1, (Patras, Greece), pp. 191–194, oct 2005.

[8] A. Meng, P. Ahrendt, and J. Larsen, "Improving music genre classification by short-time feature integration," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. V, pp. 497–500, mar 2005.

[9] H. Meinedo and J. Neto, "Audio segmentation, classification and clustering in a broadcast news task," in *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP'03)*, vol. 2, pp. 5–8, IEEE, 2003.

Appendix B

Pruning Figures

Figure B.1-B.10 in this appendix contain figures obtained from each run of the pruning-Step1 algorithm. The upper part of each figure show the best test misclassification in each iteration of the algorithm. The middle part of each figure show the features used in the model giving the best test misclassification in each iteration. White means that the feature is used, black means that the feature is pruned out. When a feature is pruned out in three succeeding iterations, the feature is deleted from the feature set and not considered in the remaining iterations. The lower part of each feature show the number of hidden units in the NN in each iteration.

Table B.1 shows the order the features are pruned out in each of the 20 run of the pruning-Step2 algorithm. The features to the left are the features pruned out first, and the features to the right is the last features deleted in the pruning scheme.

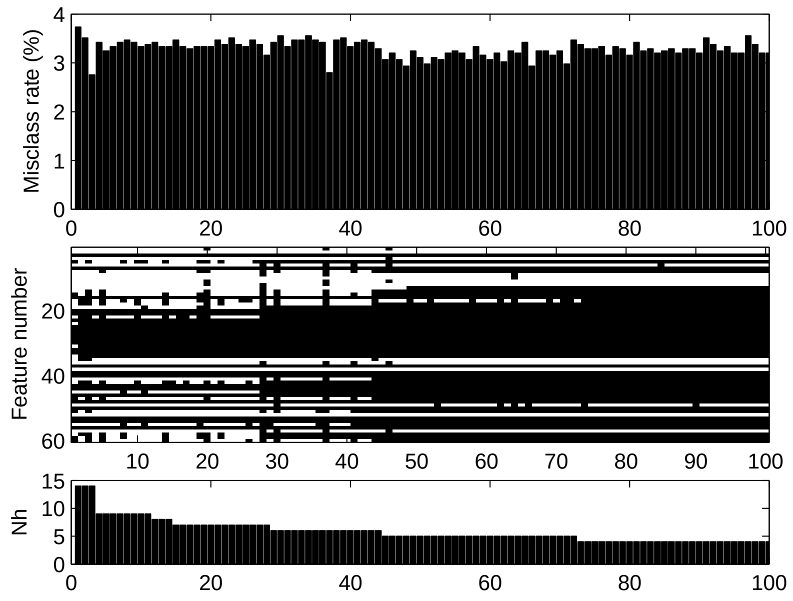


Figure B.1: pruningStep1, run1

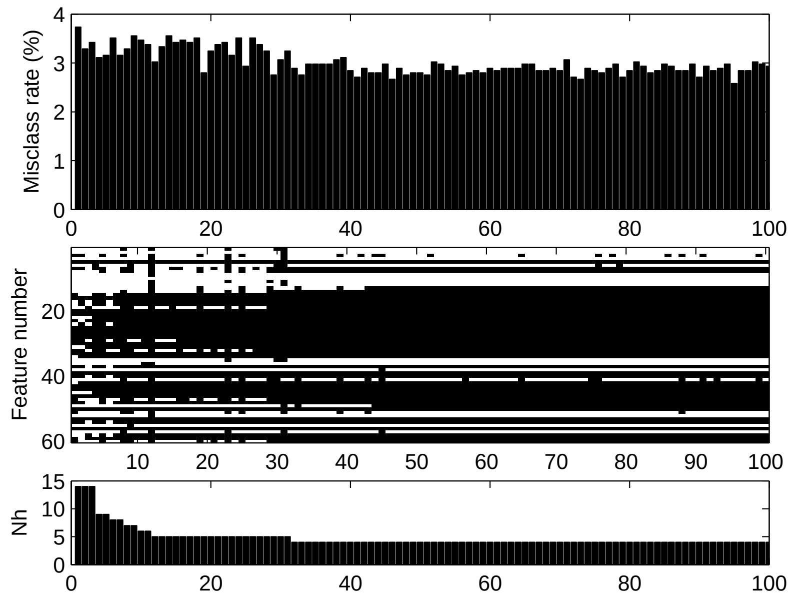


Figure B.2: pruningStep1, run2

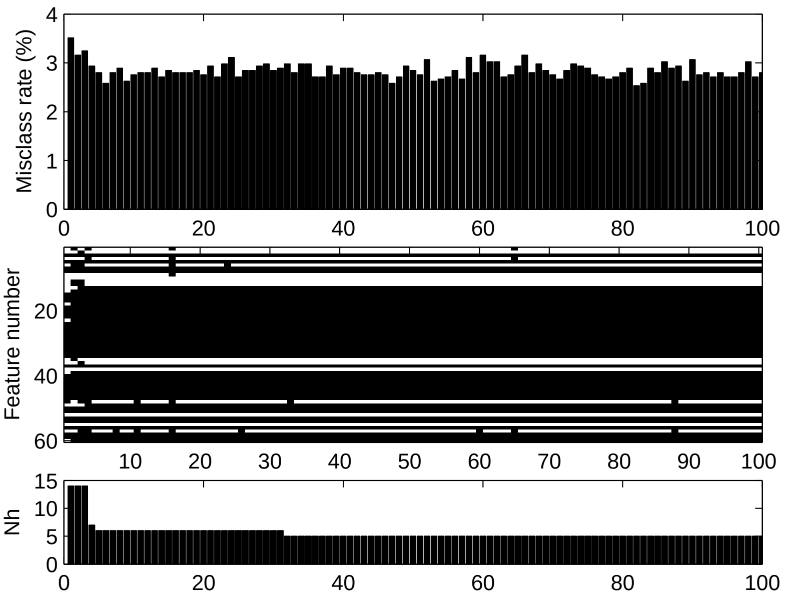


Figure B.3: PRUNINGSTEP1, run3

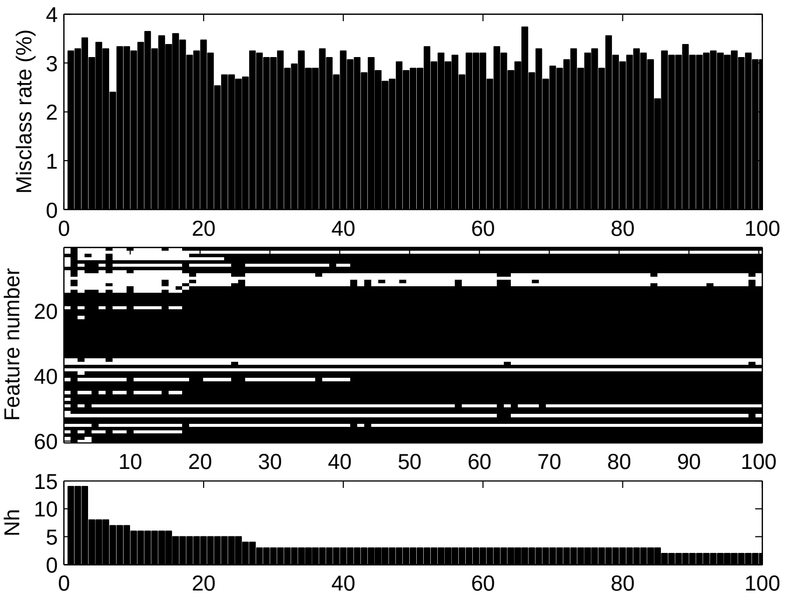


Figure B.4: PRUNINGSTEP1, run4

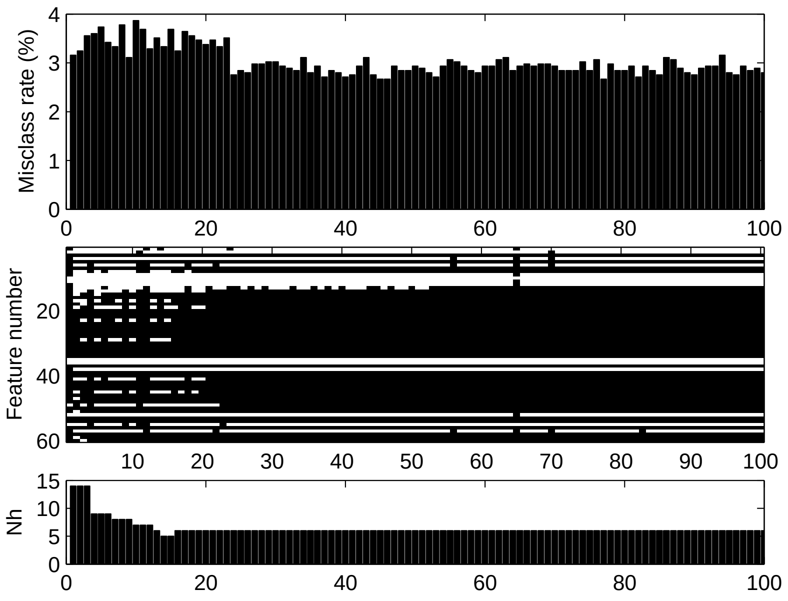


Figure B.5: PRUNINGSTEP1, run5

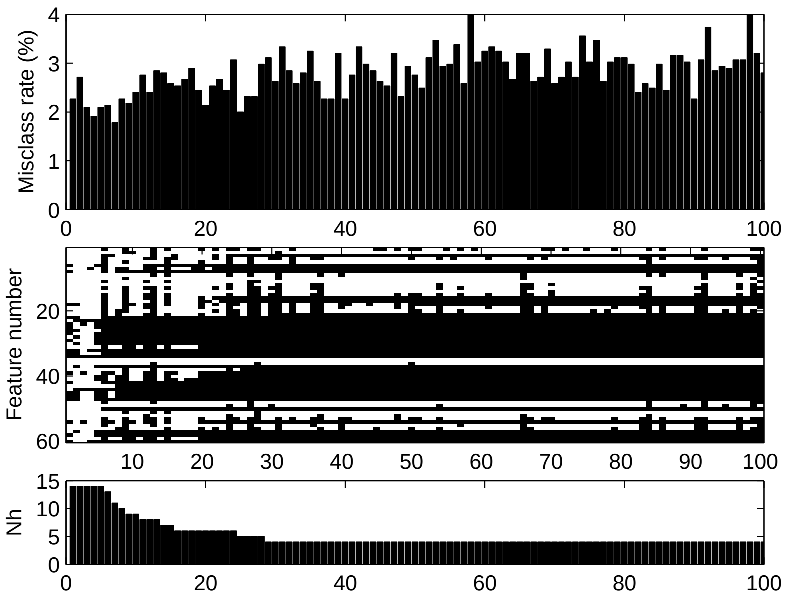


Figure B.6: PRUNINGSTEP1, run6

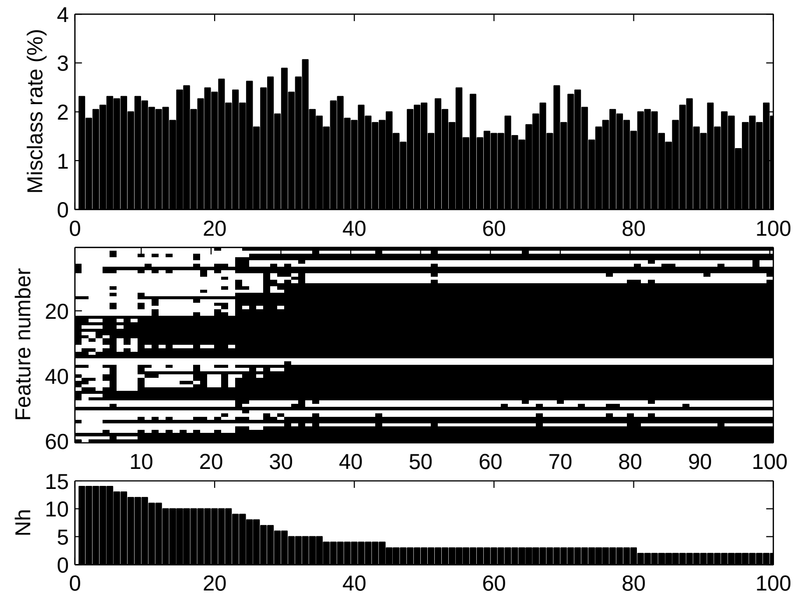


Figure B.7: pruningStep1, run7

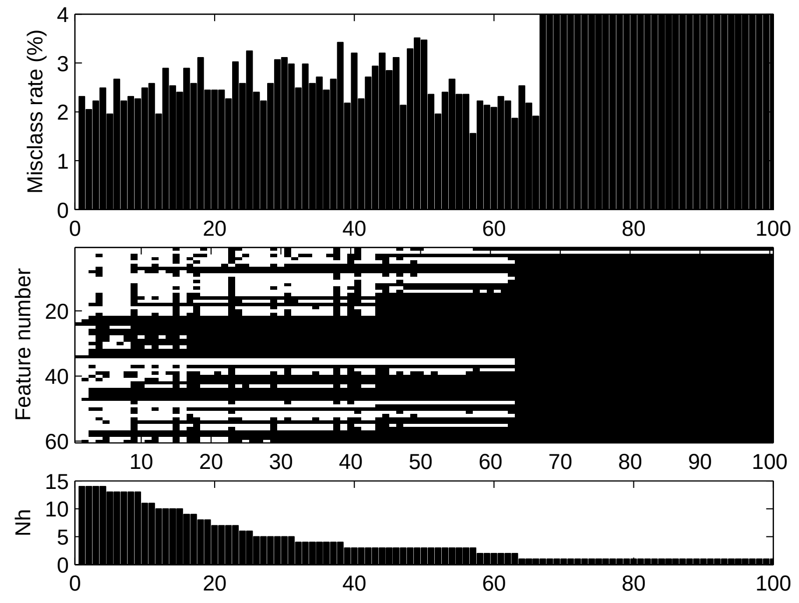


Figure B.8: pruningStep1, run8

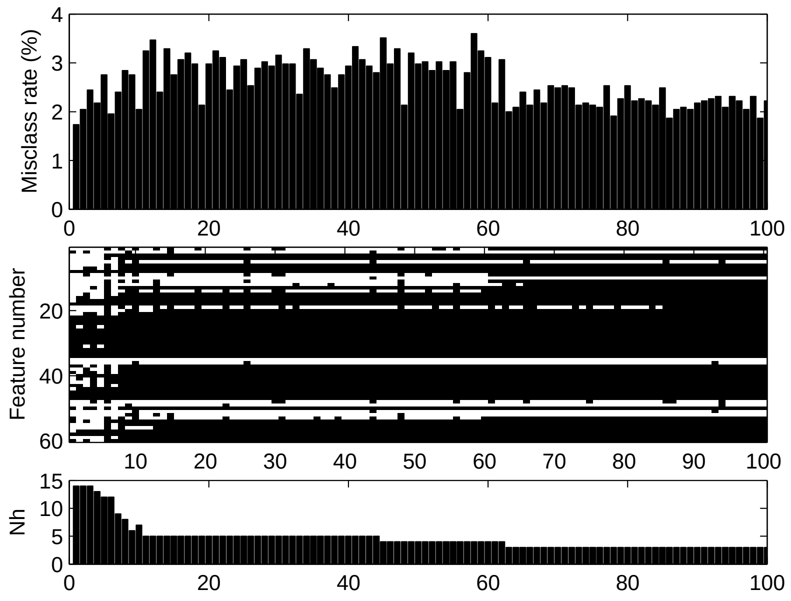


Figure B.9: pruningStep1, run9

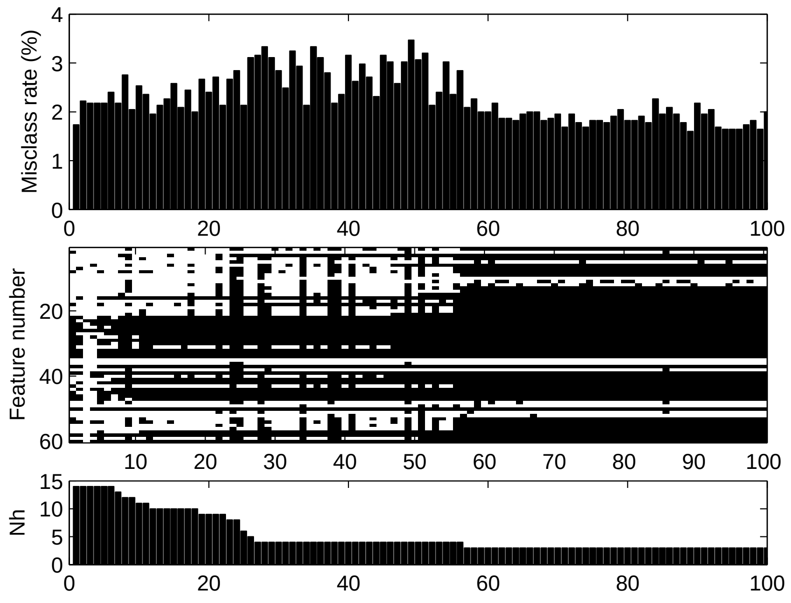


Figure B.10: pruningStep1, run10

Run 1	14	5	1	6	48	3	4	57	49	9	55	20	12	38	41	11	52	10	2	35	36	51
Run 2	52	14	5	6	4	3	20	48	38	9	12	1	41	11	49	55	57	2	10	35	36	51
Run 3	11	6	3	38	14	35	51	12	5	4	20	36	1	52	57	55	48	2	10	9	41	49
Run 4	3	48	4	1	38	49	14	6	57	12	9	20	5	52	11	55	41	2	10	35	36	51
Run 5	20	49	14	5	6	3	4	52	9	12	1	48	38	57	11	41	55	2	10	35	36	51
Run 6	51	3	49	14	20	5	4	48	6	12	9	41	1	57	52	11	10	55	2	35	36	38
Run 7	48	14	12	41	3	38	52	49	6	35	20	5	57	1	4	11	2	9	55	10	36	51
Run 8	36	1	14	3	5	38	51	57	11	48	55	4	6	9	12	20	41	10	2	35	49	52
Run 9	14	38	4	5	3	51	6	57	1	48	12	9	11	41	36	55	20	10	2	35	49	52
Run 10	12	14	52	3	5	6	9	4	38	20	41	49	48	1	57	55	11	10	2	35	36	51
Run 11	14	35	51	49	3	5	6	20	4	1	48	55	57	41	2	12	11	36	38	10	9	52
Run 12	14	48	1	5	51	20	4	41	49	12	35	6	2	57	3	11	55	36	10	38	9	52
Run 13	3	14	10	5	20	12	49	9	41	48	2	55	38	35	57	36	52	11	1	4	6	51
Run 14	57	14	10	3	20	55	1	12	2	5	49	9	48	41	11	38	4	36	51	6	35	52
Run 15	1	20	2	10	14	41	5	3	12	49	48	55	9	57	35	51	38	4	11	6	36	52
Run 16	12	38	4	9	55	51	20	3	36	11	49	41	14	48	57	5	2	1	6	10	35	52
Run 17	14	5	35	20	41	38	1	2	12	6	4	3	36	48	11	55	57	51	10	49	9	52
Run 18	55	20	5	51	14	10	38	6	35	49	12	9	36	11	57	41	4	1	2	3	48	52
Run 19	11	36	49	6	35	4	14	20	2	57	48	3	5	51	41	12	55	1	38	10	9	52
Run 20	14	5	20	36	48	51	35	41	3	4	38	6	11	57	2	12	1	55	9	10	49	52

Table B.1: Results of the PRUNINGSTEP2. The table shows the order the features were pruned out in each of the 20 runs of the algorithm. The features at the right are the features deleted last in the pruning process.

Appendix C

Kullback-Leibler Derivation

The *directed divergence* or *Kullback-Leibler number* for class ω_1 versus class ω_2 is given by:

$$\begin{aligned} I(1,2) &= \int p_1(\mathbf{x}) \ln \frac{p_1(\mathbf{x})}{p_2(\mathbf{x})} d\mathbf{x} \\ &= \int p_1(\mathbf{x}) \ln p_1(\mathbf{x}) - \int p_1(\mathbf{x}) \ln p_2(\mathbf{x}) d\mathbf{x} \end{aligned}$$

Assuming that the distributions p_i are n-variate normal distributions, i.e. $p_1((x)) \sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) p_2((x)) \sim \mathcal{N}(\boldsymbol{\mu}_2, \Sigma_2)$, we get:

$$\begin{aligned} \int p_1(\mathbf{x}) \ln p_2(\mathbf{x}) d\mathbf{x} &= \int p_1(\mathbf{x}) \ln \frac{1}{(2\pi)^{n/2} |\boldsymbol{\Sigma}_2|^{1/2}} \exp\Big\{ -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}_2^{-1} (\mathbf{x}-\boldsymbol{\mu}_2) \Big\} d\mathbf{x} \\ &= -\frac{1}{2} \ln |(2\pi)^n \boldsymbol{\Sigma}_2| - \frac{1}{2} \int p_1(\mathbf{x}) \underbrace{(\mathbf{x}-\boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}_2^{-1} (\mathbf{x}-\boldsymbol{\mu}_2)} d\mathbf{x} \end{aligned}$$

Using $\boldsymbol{\delta} = (\mathbf{x}-\boldsymbol{\mu}_2)$

$$\begin{aligned} \sum_{j,j'} (\boldsymbol{\delta}_j^\top \boldsymbol{\Sigma}_{j,j'} \boldsymbol{\delta}_{j'}) &= \sum_{j,j'} (\boldsymbol{\Sigma}_{j,j'} \boldsymbol{\delta}_j \boldsymbol{\delta}_{j'}^\top) \\ &= \sum_{j,j'} (\boldsymbol{\Sigma}_{j,j'} \mathbf{A}_{j,j'}) \\ &= \sum_{j,j'} (\mathbf{M}_{j,j'}), (\mathbf{M} = \boldsymbol{\Sigma}\mathbf{A}) \\ &= \mathrm{Tr}(\mathbf{M}_{j,j'}) \end{aligned}$$

$$\begin{aligned} \int p_1(\mathbf{x}) \ln p_2(\mathbf{x}) d\mathbf{x} &= -\frac{1}{2} \ln |(2\pi)^n \boldsymbol{\Sigma}_2| - \frac{1}{2} \int p_1(\mathbf{x}) \mathrm{Tr}(\boldsymbol{\Sigma}_2^{-1} (\mathbf{x}-\boldsymbol{\mu}_2)(\mathbf{x}-\boldsymbol{\mu}_2)^\top) d\mathbf{x} \\ &= -\frac{1}{2} \ln |(2\pi)^n \boldsymbol{\Sigma}_2| - \frac{1}{2} \mathrm{Tr}\Big[\boldsymbol{\Sigma}_2^{-1} \underbrace{\int p_1(\mathbf{x}) (\mathbf{x}-\boldsymbol{\mu}_2)^\top (\mathbf{x}-\boldsymbol{\mu}_2)\, d\mathbf{x}} \Big] \end{aligned}$$

$$\begin{aligned}
\int p_1(\mathbf{x})(\mathbf{x}-\boldsymbol{\mu}_2)(\mathbf{x}-\boldsymbol{\mu}_2)^\top d\mathbf{x} &= \int p_1(\mathbf{x})((\mathbf{x}-\boldsymbol{\mu}_1)+(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)) \\
&\quad ((\mathbf{x}-\boldsymbol{\mu}_1)+(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2))^\top d\mathbf{x} \\
&= \int p_1(\mathbf{x})\Big((\mathbf{x}-\boldsymbol{\mu}_1)(\mathbf{x}-\boldsymbol{\mu}_1)^\top \\
&\quad +(\mathbf{x}-\boldsymbol{\mu}_1)(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^\top + (\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^\top(\mathbf{x}-\boldsymbol{\mu}_1) \\
&\quad +(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^\top\Big)d\mathbf{x} \\
&= \boldsymbol{\Sigma}_1 \\
&\quad +\big\langle(\mathbf{x}-\boldsymbol{\mu}_1)(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^\top + (\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^\top(\mathbf{x}-\boldsymbol{\mu}_1)\big\rangle \\
&\quad +(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^\top \\
&= \boldsymbol{\Sigma}_1 + (\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^\top
\end{aligned}$$

$$\begin{aligned}
\int p_1(\mathbf{x})\ln p_2(\mathbf{x})d\mathbf{x} &= -\frac{1}{2}\ln|(2\pi)^n\boldsymbol{\Sigma}_2| - \frac{1}{2}\mathrm{Tr}\Big[\boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\Sigma}_1+(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^\top)\Big] \\
&= -\frac{1}{2}\ln|(2\pi)^n\boldsymbol{\Sigma}_2| - \frac{1}{2}\mathrm{Tr}\Big[\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1\Big] - \frac{1}{2}\mathrm{Tr}\Big[\boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^\top\Big]
\end{aligned}$$

$$\begin{aligned}
I(1,2) &= \int p_1(\mathbf{x})\ln p_1(\mathbf{x})d\mathbf{x} - \int p_1(\mathbf{x})\ln p_2(\mathbf{x})d\mathbf{x} \\
&= -\frac{1}{2}\ln|(2\pi)^n\boldsymbol{\Sigma}_1| - \frac{1}{2}\mathrm{Tr}\Big[\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_1\Big] \\
&\quad +\frac{1}{2}\ln|(2\pi)^n\boldsymbol{\Sigma}_2| + \frac{1}{2}\mathrm{Tr}\Big[\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1\Big] + \frac{1}{2}\mathrm{Tr}\Big[\boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^\top\Big] \\
&= \frac{1}{2}\ln\frac{|\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|} + \frac{1}{2}\mathrm{Tr}\Big[\boldsymbol{\Sigma}_1(\boldsymbol{\Sigma}_2^{-1}-\boldsymbol{\Sigma}_1^{-1})\Big] + \frac{1}{2}\mathrm{Tr}\Big[\boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^\top\Big]
\end{aligned}$$

$$\begin{aligned}
J(1,2) &= I(1,2)+I(2,1) \\
&= \frac{1}{2}\ln\frac{|\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|} + \frac{1}{2}\mathrm{Tr}\Big[\boldsymbol{\Sigma}_1(\boldsymbol{\Sigma}_2^{-1}-\boldsymbol{\Sigma}_1^{-1})\Big] + \frac{1}{2}\mathrm{Tr}\Big[\boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^\top\Big] \\
&\quad +\frac{1}{2}\ln\frac{|\boldsymbol{\Sigma}_1|}{|\boldsymbol{\Sigma}_2|} + \frac{1}{2}\mathrm{Tr}\Big[\boldsymbol{\Sigma}_2(\boldsymbol{\Sigma}_1^{-1}-\boldsymbol{\Sigma}_2^{-1})\Big] + \frac{1}{2}\mathrm{Tr}\Big[\boldsymbol{\Sigma}_1^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)^\top\Big] \\
&= \frac{1}{2}\mathrm{Tr}\Big[(\boldsymbol{\Sigma}_1-\boldsymbol{\Sigma}_2)(\boldsymbol{\Sigma}_2^{-1}-\boldsymbol{\Sigma}_1^{-1})\Big] \\
&\quad +\frac{1}{2}\mathrm{Tr}\Big[\boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^\top\Big] + \frac{1}{2}\mathrm{Tr}\Big[\boldsymbol{\Sigma}_1^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)^\top\Big] \\
&= \frac{1}{2}\mathrm{Tr}\Big[(\boldsymbol{\Sigma}_1-\boldsymbol{\Sigma}_2)(\boldsymbol{\Sigma}_2^{-1}-\boldsymbol{\Sigma}_1^{-1})\Big] \\
&\quad +\frac{1}{2}\mathrm{Tr}\Big[(\boldsymbol{\Sigma}_1^{-1}+\boldsymbol{\Sigma}_2^{-1})(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^\top\Big]
\end{aligned}$$

Appendix D

SPHINX XML-configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ******************************************************** -->
<!--  Sphinx-4 Configuration file                              -->
<!-- ******************************************************** -->

<config>
   <!-- ******************************************************** -->
   <!-- frequently tuned properties                              -->
   <!-- ******************************************************** -->
   <property name="absoluteBeamWidth"  value="20000"/>
   <property name="relativeBeamWidth"  value="1E-80"/>
   <property name="absoluteWordBeamWidth" value="20"/>
   <property name="relativeWordBeamWidth" value="1E-30"/>
   <property name="wordInsertionProbability" value="0.01"/>
   <property name="languageWeight" value="6.0"/>
   <property name="silenceInsertionProbability" value=".005"/>
   <property name="frontend" value="epFrontEnd"/>
   <property name="recognizer" value="recognizer"/>
   <property name="showCreations" value="false"/>
   <config>
      <property name="logLevel" value="SEVERE"/>
   </config>

   <!-- ************************************************** -->
   <!-- Batch mode                                          -->
   <!-- ************************************************** -->
   <component name="batch" type="edu.cmu.sphinx.tools.batch.BatchModeRecognizer">
      <propertylist name="inputDataProcessors">
         <item>streamDataSource</item>
      </propertylist>
      <property name="skip" value="0"/>
      <property name="recognizer" value="${recognizer}"/>
   </component>

   <!-- ******************************************************** -->
   <!-- word recognizer configuration                            -->
   <!-- ******************************************************** -->
   <component name="recognizer" type="edu.cmu.sphinx.recognizer.Recognizer">
      <property name="decoder" value="decoder"/>
      <propertylist name="monitors">
      <item>accuracyTracker </item>
      <item>speedTracker </item>
```

```
    <item>memoryTracker </item>
    <item>recognizerMonitor </item>
    </propertylist>
</component>

<!-- ******************************************************** -->
<!-- The Decoder configuration                                -->
<!-- ******************************************************** -->
 <component name="decoder" type="edu.cmu.sphinx.decoder.Decoder">
      <property name="searchManager" value="wordPruningSearchManager"/>
      <property name="featureBlockSize" value="10"/>
 </component>

<!-- ******************************************************** -->
<!-- The Search Manager                                       -->
<!-- ******************************************************** -->
<component name="wordPruningSearchManager" type="edu.cmu.sphinx.decoder.search.↘
     →WordPruningBreadthFirstSearchManager">
    <property name="logMath" value="logMath"/>
    <property name="linguist" value="lexTreeLinguist"/>
    <property name="pruner" value="trivialPruner"/>
    <property name="scorer" value="threadedScorer"/>
    <property name="activeListManager" value="activeListManager"/>
    <property name="growSkipInterval" value="0"/>
    <property name="checkStateOrder" value="false"/>
    <property name="buildWordLattice" value="false"/>
    <property name="maxLatticeEdges" value="3"/>
    <property name="acousticLookaheadFrames" value="1.7"/>
    <property name="relativeBeamWidth" value="${relativeBeamWidth}"/>
</component>

<!-- ******************************************************** -->
<!-- The Active Lists                                         -->
<!-- ******************************************************** -->
<component name="activeListManager" type="edu.cmu.sphinx.decoder.search.↘
     →SimpleActiveListManager">
    <propertylist name="activeListFactories">
        <item>standardActiveListFactory</item>
        <item>wordActiveListFactory</item>
        <item>wordActiveListFactory</item>
        <item>standardActiveListFactory</item>
        <item>standardActiveListFactory</item>
        <item>standardActiveListFactory</item>
    </propertylist>
</component>

<component name="standardActiveListFactory" type="edu.cmu.sphinx.decoder.search.↘
     →PartitionActiveListFactory">
    <property name="logMath" value="logMath"/>
    <property name="absoluteBeamWidth" value="${absoluteBeamWidth}"/>
    <property name="relativeBeamWidth" value="${relativeBeamWidth}"/>
</component>

<component name="wordActiveListFactory" type="edu.cmu.sphinx.decoder.search.↘
     →PartitionActiveListFactory">
    <property name="logMath" value="logMath"/>
    <property name="absoluteBeamWidth" value="${absoluteWordBeamWidth}"/>
    <property name="relativeBeamWidth" value="${relativeWordBeamWidth}"/>
</component>

<!-- ******************************************************** -->
<!-- The Pruner                                               -->
<!-- ******************************************************** -->
<component name="trivialPruner" type="edu.cmu.sphinx.decoder.pruner.SimplePruner"/>

<!-- ******************************************************** -->
<!-- The Scorer                                               -->
<!-- ******************************************************** -->
<component name="threadedScorer" type="edu.cmu.sphinx.decoder.scorer.↘
     →ThreadedAcousticScorer">
    <property name="frontend" value="${frontend}"/>
    <property name="isCpuRelative" value="false"/>
    <property name="numThreads" value="10"/>
    <property name="minScoreablesPerThread" value="10"/>
    <property name="scoreablesKeepFeature" value="false"/>
</component>

<!-- ****************************************************** -->
<!-- The linguist  configuration                            -->
<!-- ****************************************************** -->
<component name="lexTreeLinguist" type="edu.cmu.sphinx.linguist.lextree.↘
     →LexTreeLinguist">
    <property name="logMath" value="logMath"/>
    <property name="acousticModel" value="hub4"/>
    <property name="languageModel" value="trigramModel"/>
    <property name="dictionary" value="dictionaryHUB4"/>
    <property name="addFillerWords" value="false"/>
    <property name="fillerInsertionProbability" value="1E-10"/>
    <property name="generateUnitStates" value="true"/>
    <property name="wantUnigramSmear" value="true"/>
    <property name="unigramSmearWeight" value="1"/>
    <property name="wordInsertionProbability" value="${wordInsertionProbability}"/>
    <property name="silenceInsertionProbability" value="${silenceInsertionProbability↘
         →}"/>
    <property name="languageWeight" value="${languageWeight}"/>
    <property name="unitManager" value="unitManager"/>
</component>

<!-- ****************************************************** -->
<!-- The Dictionary configuration                           -->
<!-- ****************************************************** -->
<component name="dictionary" type="edu.cmu.sphinx.linguist.dictionary.FullDictionary↘
     →">
    <property name="dictionaryPath" value="resource:/edu.cmu.sphinx.model.acoustic.↘
         →WSJ_8gau_13dCep_16k_40mel_130Hz_6800Hz.Model!/edu/cmu/sphinx/model/acoustic/↘
         →WSJ_8gau_13dCep_16k_40mel_130Hz_6800Hz/dict/cmudict.0.6d"/>
    <property name="fillerPath" value="resource:/edu.cmu.sphinx.model.acoustic.↘
         →WSJ_8gau_13dCep_16k_40mel_130Hz_6800Hz.Model!/edu/cmu/sphinx/model/acoustic/↘
         →WSJ_8gau_13dCep_16k_40mel_130Hz_6800Hz/dict/fillerdict"/>
    <property name="addSilEndingPronunciation" value="false"/>
    <property name="wordReplacement" value="&lt;sil&gt;"/>
    <property name="unitManager" value="unitManager"/>
</component>

<!-- ************************************************ -->
<!-- dictionary                                       -->
<!-- ************************************************ -->
<component name="dictionaryHUB4"
      type="edu.cmu.sphinx.linguist.dictionary.FullDictionary">
    <property name="dictionaryPath" value="resource:/edu.cmu.sphinx.model.acoustic.↘
         →HUB4_8gau_13dCep_16k_40mel_133Hz_6855Hz.Model!/edu/cmu/sphinx/model/acoustic↘
         →/HUB4_8gau_13dCep_16k_40mel_133Hz_6855Hz/cmudict.06d"/>
    <property name="fillerPath" value="resource:/edu.cmu.sphinx.model.acoustic.↘
         →HUB4_8gau_13dCep_16k_40mel_133Hz_6855Hz.Model!/edu/cmu/sphinx/model/acoustic↘
         →/HUB4_8gau_13dCep_16k_40mel_133Hz_6855Hz/fillerdict"/>
    <property name="addSilEndingPronunciation" value="false"/>
    <property name="allowMissingWords" value="false"/>
    <property name="unitManager" value="unitManager"/>
</component>

<!-- ****************************************************** -->
<!-- The Language Model configuration                       -->
<!-- ****************************************************** -->
<component name="trigramModel" type="edu.cmu.sphinx.linguist.language.ngram.large.↘
     →LargeTrigramModel">
    <property name="unigramWeight" value=".5"/>
    <property name="maxDepth" value="3"/>
    <property name="logMath" value="logMath"/>
    <property name="dictionary" value="dictionaryHUB4"/>
    <property name="location" value="/gbar/erlang/home1/s00/s001498/pep/sphinx4/lib/↘
         →hub4/language_model.arpaformat.DMP"/>
</component>

<!-- ****************************************************** -->
<!-- The acoustic model configuration                       -->
<!-- ****************************************************** -->
<component name="wsj" type="edu.cmu.sphinx.model.acoustic.↘
     →WSJ_8gau_13dCep_16k_40mel_130Hz_6800Hz.Model">
    <property name="loader" value="wsjLoader"/>
    <property name="unitManager" value="unitManager"/>
</component>

<component name="wsjLoader" type="edu.cmu.sphinx.model.acoustic.↘
    →WSJ_8gau_13dCep_16k_40mel_130Hz_6800Hz.ModelLoader">
    <property name="logMath" value="logMath"/>
    <property name="unitManager" value="unitManager"/>
</component>

<!-- ************************************************** -->
<!-- HUB4                                               -->
<!-- ************************************************** -->
<component name="hub4" type="edu.cmu.sphinx.model.acoustic.↘
    →HUB4_8gau_13dCep_16k_40mel_133Hz_6855Hz.Model">
    <property name="loader" value="sphinx3Loader"/>
    <property name="unitManager" value="unitManager"/>
</component>

<component name="sphinx3Loader" type="edu.cmu.sphinx.model.acoustic.↘
    →HUB4_8gau_13dCep_16k_40mel_133Hz_6855Hz.ModelLoader">
    <property name="logMath" value="logMath"/>
    <property name="unitManager" value="unitManager"/>
</component>

<!-- ****************************************************** -->
<!-- The unit manager configuration                         -->
<!-- ****************************************************** -->
<component name="unitManager" type="edu.cmu.sphinx.linguist.acoustic.UnitManager"/>

<!-- ****************************************************** -->
<!-- The frontend configuration                             -->
<!-- ****************************************************** -->
<component name="epFrontEnd" type="edu.cmu.sphinx.frontend.FrontEnd">
    <propertylist name="pipeline">
        <item>streamDataSource </item>
        <item>speechClassifier </item>
        <item>speechMarker </item>
        <item>nonSpeechDataFilter </item>
        <item>premphasizer </item>
        <item>windower </item>
        <item>fft </item>
        <item>melFilterBank </item>
        <item>dct </item>
        <item>liveCMN </item> <!-- batchCMN for batch mode -->
        <item>featureExtraction </item>
    </propertylist>
</component>

<component name="streamDataSource" type="edu.cmu.sphinx.frontend.util.↘
    →StreamDataSource">
    <property name="sampleRate" value="16000"/>
    <property name="bitsPerSample" value="16"/>
    <property name="bigEndianData" value="false"/>
    <property name="signedData" value="true"/>
    <property name="bytesPerRead" value="320"/>
</component>

<component name="microphone" type="edu.cmu.sphinx.frontend.util.Microphone">
    <property name="closeBetweenUtterances" value="false"/>
</component>

<component name="speechClassifier" type="edu.cmu.sphinx.frontend.endpoint.↘
    →SpeechClassifier">
    <property name="threshold" value="10"/>
    <property name="debug" value="false"/>
</component>

<component name="nonSpeechDataFilter" type="edu.cmu.sphinx.frontend.endpoint.↘
    →NonSpeechDataFilter">
    <property name="mergeSpeechSegments" value="true"/>
</component>

<component name="speechMarker" type="edu.cmu.sphinx.frontend.endpoint.SpeechMarker">
```

```
      <property name="speechTrailer" value="50"/>
   </component>

   <component name="premphasizer" type="edu.cmu.sphinx.frontend.filter.Preemphasizer"/>

   <component name="windower" type="edu.cmu.sphinx.frontend.window.RaisedCosineWindower↘
       →"/>

   <component name="fft" type="edu.cmu.sphinx.frontend.transform.↘
       →DiscreteFourierTransform"/>

   <component name="melFilterBank" type="edu.cmu.sphinx.frontend.frequencywarp.↘
       →MelFrequencyFilterBank"/>

   <component name="dct" type="edu.cmu.sphinx.frontend.transform.↘
       →DiscreteCosineTransform"/>

   <component name="liveCMN" type="edu.cmu.sphinx.frontend.feature.LiveCMN"/>

   <component name="batchCMN" type="edu.cmu.sphinx.frontend.feature.BatchCMN"/>

   <component name="featureExtraction" type="edu.cmu.sphinx.frontend.feature.↘
       →DeltasFeatureExtractor"/>

   <!-- ******************************************************* -->
   <!--   monitors                                              -->
   <!-- ******************************************************* -->
   <component name="accuracyTracker" type="edu.cmu.sphinx.instrumentation.↘
       →BestConfidenceAccuracyTracker">
      <property name="confidenceScorer" value="confidenceScorer"/>
      <property name="recognizer" value="${recognizer}"/>
      <property name="showRawResults" value="true"/>
      <property name="showAlignedResults" value="true"/>
   </component>

   <component name="confidenceScorer" type="edu.cmu.sphinx.result.SausageMaker"/>

   <component name="memoryTracker" type="edu.cmu.sphinx.instrumentation.MemoryTracker">
      <property name="recognizer" value="${recognizer}"/>
      <property name="showDetails" value="false"/>
      <property name="showSummary" value="false"/>
   </component>

   <component name="speedTracker" type="edu.cmu.sphinx.instrumentation.SpeedTracker">
      <property name="recognizer" value="${recognizer}"/>
      <property name="frontend" value="${frontend}"/>
      <property name="showDetails" value="false"/>
   </component>

   <component name="recognizerMonitor" type="edu.cmu.sphinx.instrumentation.↘
       →RecognizerMonitor">
      <property name="recognizer" value="${recognizer}"/>
      <propertylist name="allocatedMonitors">
         <item>configMonitor </item>
      </propertylist>
   </component>

   <component name="configMonitor" type="edu.cmu.sphinx.instrumentation.ConfigMonitor">
      <property name="showConfig" value="false"/>
   </component>

   <component name="logMath" type="edu.cmu.sphinx.util.LogMath">
      <property name="logBase" value="1.0001"/>
      <property name="useAddTable" value="true"/>
   </component>
</config>
```

Appendix E

Speech Recognition Results

This appendix lists the result from SPHINX-4 using the CNN example described in section 6.2. Each speaker segment is listed separately.

The listings contain the following:

REF is the true transcriptions (reference).

HYP is the output from the ASR (hypothesis).

ALIGN_REF is the reference where substitutions (wrong words) are written with capital letters, and deletions (missing words) are marked with stars (*).

ALIGN_HYP is the hypothesis where substitutions are written with capital letters and insertions (inserted words) are marked with stars.

The ALIGN_REF and ALIGN_HYP are aligned, thus it is easy to see the difference between the true transcript and the output from ASR.

E.1 Segment 1

```
REF:       c. n. n. radio i'm jim ribble saddam husseins defense team walked
out of the courtroom today after a heated argument over the legitimacy of the
trial after some deliberation the judge allowed them back in to resume
proceedings one issue was over whether former u. s. attorney general ramsey
clark who was helping the defense team would be allowed to speak eventually
he was allowed

HYP:       c. n. n. radio and jim rubble so i was sainz defense team walked
out the courtroom today after a heated argument over the legitimacy of the
trial after some deliberation the judge allowed them back into resumed
proceedings one issue was over whether former u. s. attorney general ramsey
clark who was hoping defense team would be allowed to speak eventually
he was allowed



ALIGN_REF: c. n. n. radio I'M jim ****** ** RIBBLE SADDAM HUSSEINS defense
ALIGN_HYP: c. n. n. radio AND jim RUBBLE SO I      WAS    SAINZ    defense

team walked out OF the courtroom today after a heated argument over the
team walked out ** the courtroom today after a heated argument over the

legitimacy of the trial after some deliberation the judge allowed them back
legitimacy of the trial after some deliberation the judge allowed them back

IN   TO       RESUME proceedings one issue was over whether former u. s.
INTO RESUMED  ****** proceedings one issue was over whether former u. s.

attorney general ramsey clark who was HELPING THE defense team would be
attorney general ramsey clark who was HOPING  *** defense team would be

allowed to speak eventually he was allowed
allowed to speak eventually he was allowed


   Accuracy: 84.848%    Errors: 12  (Sub: 7  Ins: 2  Del: 3)
   Words: 66   Matches: 56    WER: 18.182%
```

E.2 Segment 2

```
REF:       this trial can either divide or heal and unless it is seen as
absolutely fair and is absolutely fair in fact it will irreconcilably divide
the people of iraq

HYP:       trial either divide or he'll let fishing as after the ferron is
absolutely fair in fact ewing irreconcilable a divide people of iraq



ALIGN_REF: THIS trial CAN either divide or HEAL  AND UNLESS  IT IS SEEN as
ALIGN_HYP: **** trial *** either divide or HE'LL LET FISHING ** ** **** as

ABSOLUTELY FAIR AND    is absolutely fair in fact IT    WILL
AFTER      THE  FERRON is absolutely fair in fact EWING IRRECONCILABLE

IRRECONCILABLY divide THE people of iraq
A              divide *** people of iraq


   Accuracy: 48.276%    Errors: 15  (Sub: 9  Ins: 0  Del: 6)
   Words: 29   Matches: 14    WER: 51.724%
```

E.3 Segment 3

```
REF:       at one point hussein himself stood up and shook his fist
shouting long live iraq the nine eleven commission releases a new report
today assessing how well the government is responded to its recommendations
for making the country safer correspondent dick uliano reports it is expected
to handout low marks

HYP:       or one point hussein and soul stood up and shook his fist
chopping long live iraq final elam commission lisa's a new report
today assessing how well the government is responded to its recommendations
for making the country safer correspondent achille all reports his expected
to hand out some lobel arcs



ALIGN_REF: AT one point hussein *** HIMSELF stood up and shook his fist
ALIGN_HYP: OR one point hussein AND SOUL    stood up and shook his fist

SHOUTING long live iraq THE   NINE ELEVEN commission RELEASES a new report
CHOPPING long live iraq FINAL ELAM ****** commission LISA'S   a new report

today assessing how well the government is responded to its recommendations
today assessing how well the government is responded to its recommendations

for making the country safer correspondent DICK    ULIANO reports IT  IS
for making the country safer correspondent ACHILLE ALL    reports HIS **

expected to **** *** HANDOUT LOW   MARKS
expected to HAND OUT SOME    LOBEL ARCS


   Accuracy: 72.000%    Errors: 17  (Sub: 12  Ins: 3  Del: 2)
   Words: 50   Matches: 36    WER: 34.000%
```

E.4 Segment 4

```
REF:       former nine eleven commission leaders lee hamilton and thomas
kane say president bush in congress deserve failing grades for failing
to follow the commission's recommendations to make the nation safer against
another nine eleven style attack lee hamilton telling meet the press

HYP:       former eleven commission leaders lee hamilton and thomas
kane say president bush in congress deserve failing grades for failing
to follow the commission's recommendations to make a nation safer against
another not olefins style attack lee hamilton telling meet the press



ALIGN_REF: former NINE eleven commission leaders lee hamilton and thomas
ALIGN_HYP: former **** eleven commission leaders lee hamilton and thomas

kane say president bush in congress deserve failing grades for failing to
kane say president bush in congress deserve failing grades for failing to

follow the commission's recommendations to make THE nation safer against
follow the commission's recommendations to make A   nation safer against

another NINE ELEVEN  style attack lee hamilton telling meet the press
another NOT  OLEFINS style attack lee hamilton telling meet the press


   Accuracy: 90.476%    Errors: 4  (Sub: 3  Ins: 0  Del: 1)
   Words: 42   Matches: 38    WER: 9.524%
```

E.5 Segment 5

```
REF:       there is a lack of sense of urgency and that’s what
impresses us overall

HYP:       or zaid lack of a sense of urgency and that’s what
impresses us overall



ALIGN_REF: THERE IS   A lack of * sense of urgency and that’s what
ALIGN_HYP: OR    ZAID * lack of A sense of urgency and that’s what

impresses us overall
impresses us overall


   Accuracy: 78.571%    Errors: 4  (Sub: 2  Ins: 1  Del: 1)
   Words: 14   Matches: 11    WER: 28.571%
```

E.6 Segment 6

```
REF:       the white house’s changes are being implemented we’re safer
but not yet safe but hamilton and kane say first responders still unable
to communicate on the same radio frequencies homeland security money being
foolishly spent and slow going securing the nation’s nuclear facilities
reporting live dick uliano c. n. n. washington

HYP:       the white house’s changes are being implemented or safer
but not yet safe but hamilton taints a first responder still are unable
i communicate on the same radio frequencies public security money being
foolishly spent and slow going securing the nation’s nuclear facilities
reporting live to kill iago c. n. n. washington



ALIGN_REF: the white house’s changes are being implemented WE’RE safer but
ALIGN_HYP: the white house’s changes are being implemented OR    safer but

not yet safe but hamilton AND    KANE SAY first RESPONDERS still *** unable
not yet safe but hamilton TAINTS A    *** first RESPONDER  still ARE unable

TO communicate on the same radio frequencies HOMELAND security money being
I  communicate on the same radio frequencies PUBLIC   security money being

foolishly spent and slow going securing the nation’s nuclear facilities
foolishly spent and slow going securing the nation’s nuclear facilities

reporting live ** DICK ULIANO c. n. n. washington
reporting live TO KILL IAGO   c. n. n. washington


   Accuracy: 82.353%    Errors: 11  (Sub: 8  Ins: 2  Del: 1)
   Words: 51   Matches: 42    WER: 21.569%
```

E.7 Segment 7

```
REF:       a suicide bomber killed at least five people injured about three
dozen at a mall in the northern israeli city of netania correspondent john
wauss reports from israel

HYP:       a suicide bomber killed at least five people injured about three
dozen of a bald northern israeli city of titania correspondent john
paul's reports from israel



ALIGN_REF: a suicide bomber killed at least five people injured about
ALIGN_HYP: a suicide bomber killed at least five people injured about

three dozen AT a MALL IN THE northern israeli city of NETANIA correspondent
three dozen OF a BALD ** *** northern israeli city of TITANIA correspondent

john WAUSS  reports from israel
john PAUL'S reports from israel



   Accuracy: 78.571%    Errors: 6  (Sub: 4  Ins: 0  Del: 2)
   Words: 28   Matches: 22    WER: 21.429%
```

E.8 Segment 8

```
REF:       the suicide bomber waited in line outside the shopping mall
in netania tonya when security guards and police thought something was up they
asked him to step away from the crowd as he walked away he detonated his
explosives

HYP:       the suicide all white in line outside dissolving molding
that tonya insecurity godsend laced billets awning was outlay
aussie to step away from the crowd as he walked away he detonated his
explosives



ALIGN_REF: the suicide BOMBER WAITED in line outside THE        SHOPPING MALL
ALIGN_HYP: the suicide ALL    WHITE  in line outside DISSOLVING MOLDING  THAT

IN NETANIA tonya WHEN       SECURITY GUARDS AND     POLICE THOUGHT SOMETHING
** ******* tonya INSECURITY GODSEND  LACED  BILLETS AWNING ******* *********

was UP     THEY   ASKED HIM to step away from the crowd as he walked away he
was OUTLAY AUSSIE ***** *** to step away from the crowd as he walked away he

detonated his explosives
detonated his explosives


   Accuracy: 53.846%    Errors: 18  (Sub: 12  Ins: 0  Del: 6)
   Words: 39   Matches: 21    WER: 46.154%
```

E.9 Segment 9

```
REF:       the military says two us helicopters made emergency landings in
afghanistan after being hit by enemy fire five american and one afghan
soldier were reported injured none seriously the most trusted name in news
this is c.n.n. radio


HYP:       all jory says to us helicopters made emergency landings in
afghanistan after being hit by enemy fire five american one afghans
soldier were reported jerk and seriously most trusted ne menus d'souza
yen in reading



ALIGN_REF: THE MILITARY says TWO us helicopters made emergency landings in
ALIGN_HYP: ALL JORY     says TO  us helicopters made emergency landings in

afghanistan after being hit by enemy fire five american AND one AFGHAN  soldier
afghanistan after being hit by enemy fire five american *** one AFGHANS soldier

were reported INJURED NONE seriously THE most trusted NAME IN    NEWS
were reported JERK    AND  seriously *** most trusted NE   MENUS D'SOUZA

THIS IS C.N.N.  RADIO
YEN  IN READING *****


   Accuracy: 60.526%    Errors: 15  (Sub: 12  Ins: 0  Del: 3)
   Words: 38   Matches: 23    WER: 39.474%
```

Bibliography

C. M. Bishop. *Neural Networks for Pattern Recognition.* Clarendon Press, Oxford, 1995. ISBN 0198538642.

M. Brookes. Voicebox: Speech processing toolbox for matlab. Internet, 1998. URL `http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html`.

M. Cettolo, M. Vescovi, and R. Rizzi. Evaluation of bic-based algorithms for audio segmentation. *Computer Speech and Language*, 19:147–170, 2005.

J. R. Deller, J. G. Proakis, and J. H. L. Hansen. *Discrete-time Processing of Speech Signals.* Prentice Hall, New Jersey, 1993. ISBN 0023283017.

L. Feng. Speaker recognition. Master's thesis, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2004. URL `http://www2.imm.dtu.dk/pubdb/p.php?3319`. Supervised by Prof. Lars Kai Hansen.

T. Ganchev, N. Fakotakis, and G. Kokkinakis. Comparative evaluation of various mfcc implementations on the speaker verification task. In *10th International Conference on Speech and Computer, SPECOM 2005*, volume 1, pages 191–194, Patras, Greece, oct 2005.

J.-L. Gauvain, L. Lamel, and G. Adda. The limsi broadcast news transcription system. *Speech Communication*, 37(1-2):89–108, 2002. ISSN 0167-6393.

T. Hain, S. Johnson, A. Tuerk, P. C. Woodland, and S. Young. Segment generation and clustering in the htk broadcast news transcription system. In *Proc. of 1998 DARPA Broadcast News Transcription and Understanding Workshop*, pages 133–137, 1998.

T. Hain and P. C. Woodland. Segmentation and classification of broadcast news audio. In *Proc. of International Conference on Spoken Language Processing, ICSLP 1998*, volume 6, pages 2727–2730, 1998.

J. H. L. Hansen, R. Huang, B. Zhou, M. Seadle, J. R. Deller, A. R. Gurijala, M. Kurimo, and P. Angkititrakul. Speechfind: Advances in spoken document retrieval for a national gallery of the spoken word. *IEEE Transactions on Speech and Audio Processing*, 13(5): 712–730, september 2005.

R. Huang and J. H. Hansen. Advances in unsupervised audio segmentation for the broadcast news and ngsw corpora. In *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2004*, volume 1, pages 741–744, May 2004a.

R. Huang and J. H. Hansen. High-level feature weighted gmm network for audio stream classification. In *Proc. of International Conference on Spoken Language Processing, Interspeech-2004/ICSLP-2004*, volume 1, pages 1–4, Oct 2004b.

S. E. Johnson, P. Jourlin, K. S. Jones, and P. C. Woodland. Information retrieval from unsegmented broadcast news audio. *International Journal of Speech Technology*, 4:251–268, 2001.

K. W. Jørgensen and L. L. Mølgaard. Speaker recognition. Technical report, Technical University of Denmark, Informatics and Mathematical Modelling, 2005. URL `http://www2.imm.dtu.dk/pubdb/p.php?4414`.

T. Kemp, M. Schmidt, M. Westphal, and A. Waibel. Strategies for automatic segmentation of audio data. *IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings*, 3:1423–1426, 2000.

H.-G. Kim, D. Ertelt, and T. Sikora. Hybrid speaker-based segmentation system using model-level clustering. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '05)*, volume 1, pages 745–748. IEEE, 2005.

T. Kinnunen, T. Kilpeläinen, and P. Fränti. Comparison of clustering algorithms in speaker identification. In *Proceedings of the IASTED International Conference. Signal Processing and Communications*, pages 222–7. IASTED/ACTA Press, 2000.

T. Kolenda, S. Sigurdsson, O. Winther, L. K. Hansen, and J. Larsen. DTU:toolbox, 2002. URL `http://isp.imm.dtu.dk/toolbox/`.

D. Li, I. K. Sethi, N. Dimitrova, and T. McGee. Classification of general audio data for content-based retrieval. *Pattern Recognition Letters*, 22(5):533–544, 2001.

L. Lu and H. Zhang. Unsupervised speaker segmentation and tracking in real-time audio content analysis. *Multimedia Systems*, 10(Issue.4):332–343, 2005.

L. Lu, H.-J. Zhang, and H. Jiang. Content analysis for audio classification and segmentation. *IEEE Transactions on Speech and Audio Processing*, 10(7):504–516, 2002.

M. F. McKinney and J. Breebaart. Features for audio and music classification. *Proc. of ISMIR*, pages 151–158, 2003.

H. Meinedo and J. Neto. Audio segmentation, classification and clustering in a broadcast news task. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP'03)*, volume 2, pages 5–8. IEEE, 2003.

A. Meng, P. Ahrendt, and J. Larsen. Improving music genre classification by short-time feature integration. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume V, pages 497–500, mar 2005. URL `http://www2.imm.dtu.dk/pubdb/p.php?3309`.

I. Nabney and C. Bishop. Netlab toolbox. Internet, June 2004. URL `http://www.ncrg.aston.ac.uk/netlab/index.php`.

S. Nakagawa and K. Mori. Speaker change detection and speaker clustering using vq distortion measure. *Systems and Computers in Japan*, 34(13):25–35, 2003.

NIST. Rich transcription task. `http://nist.gov/speech/tests/rt/`, 2005.

B. Pellom and K. Hacioğlu. Sonic: The university of colorado continuous speech recognizer. Tech Report TR-CSLR-2001-01, Center for Spoken Language Research, University of Colorado, Boulder, 2001.

L. R. Rabiner. Tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

M. K. Ravishankar. *Efficient Algorithms for Speech Recognition*. Ph.d. thesis, School of Computer Science, Computer Science Division, Carnegie Mellon University, 1996.

J. Saunders. Real-time discrimination of broadcast speech/music. In *IEEE International Conference on Acoustics, Speech, and Signal Processing. (ICASSP'96). Conference Proceedings.*, volume 2, pages 993–996, Atlanta, GA, USA, May 1996. IEEE.

E. Scheirer and M. Slaney. Construction and evaluation of a robust multifeature speech/-music discriminator. In *Proc. of the 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '97)*, volume 2, pages 1331–1334, Washington, DC, USA, 1997. IEEE Computer Society. ISBN 0-8186-7919-0.

M. Siegler, U. Jain, B. Raj, and R. Stern. Automatic segmentation, classification and clustering of broadcast news audio. *DARPA Speech Recognition Workshop*, pages 97–99, 1997.

Sphinx-4. A speech recognizer written entirely in the java^{TM}programming language. Internet, 2004. URL `http://cmusphinx.sourceforge.net/sphinx4/`.

J.-M. Van Thong, P. Moreno, B. Logan, B. Fidler, K. Maffey, and M. Moores. Speechbot: an experimental speech-based search engine for multimedia content on the web. *IEEE Transactions on Multimedia*, 4(Issue.1):88–96, 2002.

A. Vandecatseye and J.-P. Martens. A fast, accurate and stream-based speaker segmentation and clustering algorithm. In *Proc. of 8th European Conference on Speech Communication and Technology (EUROSPEECH-2003)*, pages 941–944, 2003.

W. Walker, P. Lamere, P. Kwok, B. Raj, R. Singh, E. Gouvea, P. Wolf, and J. Woelfel. Sphinx-4: A flexible open source framework for speech recognition. Tech Report TR-2004-127, Sun Microsystems, 2004.

T. Zhang and C.-C. Jay Kuo. Audio content analysis for online audiovisual data segmentation and classification. *IEEE Transactions on Speech and Audio Processing*, 9(Issue.4): 441–457, 2001.