NTNU

Norwegian University of
Science and Technology

Indexing of Audio Databases

Event Log of Broadcast News

Ida Onshus

Master of Science in Electronics

Submission date: June 2011

Supervisor: Torbjørn Svendsen, IET

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Problem description

An increasing part of the information available on the Internet is non-textual. Examples are sound archives such as recorded lectures, net radio and archived radio transmissions, and videos such as YouTube and Internet TV. This makes a need for a tool that can search in other information sources than the text based. There is a desire to develop a tool for information search by means of sound tracks in audio databases. As a first step in this process, one needs to make an event log through audio segmentation. The segmentation includes classification of sound segments as speech or non-speech, detection of speech quality on basis of bandwidth and identification of speech segments from the same speaker. In this thesis, one wants to start creating an event log of news broadcasts, by regarding detection of long pauses and detection of speaker changes. The work will be based on a corpus from The Norwegian Broadcasting Corporation (NRK) radio.

Assignment given: 17. January 2011
Supervisor: Torbjørn Svendsen, IET.

Summary

The amount of non-textual media on the Internet is increasing, which creates a greater need of being able to search in this type of media. The goal with this thesis is to be able to do information search by use of soundtracks in audio databases. To get to know the content in an audio file, one wants a system that can automatically extract necessary information. The first step in making this system is to record what is happening at which time in an event log. This thesis treats the beginning of such a process. The experiments performed dealt with detection of pauses lasting longer than 1 second and detection of speaker changes. The corpus used in experiments consists of news broadcasts from The Norwegian Broadcasting Corporation (NRK) radio. Each broadcast had a transcription, which was used as a reference when evaluating the results. Another corpus, the HUB-4 1997 evaluation data, was used for comparative tests.

A lot of work treating indexing of audio databases has already been conducted. As corpora are different, there may be varying results obtained from the same methods. In this thesis, common segmentation methods have been used with the parameters adapted to give as good results as possible with the given corpus. In the pause detection, model-based segmentation was used. A Gaussian mixture model was implemented for each of the two events: sound and long pause. For the speaker segmentation, experiments with different metric-based segmentation techniques were performed. The Bayesian information criterion (BIC) and a modified version of this criterion[1], were tested with different options and parameter values. A false alarm compensation based on the symmetric Kullback-Leibler distance was implemented as an attempt to reduce the number of false change points.

The pause detection was not successful. By using the manual transcription as reference, an F-score of 38.1 % was obtained when the settings were adjusted to result in about the same numbers for false alarms and false rejections. However, further investigation showed that the transcription had flaws with respect to labeling of pauses. An evaluation of the wrongly in-

[1]presented by Ajmera, McCowan, and Bourlard [1]

serted pauses showed that most of these segments actually contained silence or noise. However, the number of pauses missed was unknown, and it was not possible to get a reliable F-score. An attempt on labeling all pauses in the HUB-4 1997 data was done. With the modified transcription, an F-score of 81.7 % was obtained. However, it is possible that unlabeled pauses still exist in the transcription, as the labeling was performed by only looking at the audio signal. From classification experiments it became clear that using 1st and 2nd order delta coefficients in the feature vectors gave an improvement over just using static MFCCs. An F-score of 98.8 % was obtained from these experiments, which implies that the models are good when the segment boundaries are known. In order to get trustworthy results from the recognition task, a review of the transcription must be done.

When using the modified version of BIC and false alarm compensation for speaker change detection, an F-score of 77.1 % were obtained. The average mismatch between correctly detected change points and reference transcription was 339 milliseconds. As a measure of how good the algorithm is, an F-score of 72.8 % was obtained with the HUB-4 1997 data. Ajmera et al. [1] obtained an F-score of 67 % with the same data. It became clear that full covariance matrices gave an improvement over diagonal covariance matrices, and that static MFCCs as feature vectors gave better results than MFCCs including delta coefficients. Inclusion of pitch as another feature did not contribute to any improvement of the results.

Preface

This report treats my work carried out in my final semester at the Master of Science program in Electronics, at the Department of Electronics and Telecommunications, NTNU. I would especially like to thank Professor Torbjørn Svendsen, my supervisor, for helpful support during the semester. I would also like to thank Stian Michael Kristoffersen for advice on report writing.

Trondheim, June 2, 2011
Ida Onshus

Contents

List of Figures

List of Tables

Abbreviations

A list of all abbreviations used in this thesis is given below:

BIC = Bayesian Information Criterion
CA = Correct Acceptance
CMN = Cepstral Mean Normalization
CR = Correct Rejection
DCT = Discrete Cosine Transform
EM = Expectation Maximization
FA = False Alarm
FAC = False Alarm Compensation
FFT = Fast Fourier Transform
FR = False Rejection
GMM = Gaussian Mixture Model
HMM = Hidden Markov Model
KL2 = Symmetric Kullback-Leibler
LL = Log Likelihood
ms = milliseconds
NRK = The Norwegian Broadcasting Corporation
PDF = Probability Density Function
PRC = Precision
RCL = Recall

Chapter 1

Introduction

This chapter is an introduction to the report. Section 1.1 presents the motivation and background for the work that has been done. Section 1.2 describes the chosen approach to the work, and gives some examples of previous work. Finally, section 1.3 presents the structure of the report.

1.1 Motivation and background

Non-textual media are more and more common on the Internet. Consequently there is a greater need for being able to search in this kind of information sources, in the same way as one can do with stored text. A tool for doing this can be used to search in and for e.g. answering machine messages, recorded lectures, YouTube videos, archived radio broadcast, Internet TV shows and so on. An example of previous work is the MIT lecture browser [2], which makes it possible to do searches in the recorded lectures at MIT. A screen shot of this application is shown in figure 1.1, when a search for the phrase "speech recognition" has been done. On the left hand side, all lectures that contain the phrase are listed and one can choose to listen to the specific sections where the phrase was found. On the right hand side, there is a video recording of the lecture with the spoken words displayed as text underneath. By reading one can more quickly get the context of the section containing the desired phrase.

This thesis makes the start of the development of a tool for information search by means of sound tracks in audio databases. The first step in creating this tool is to make an event log by performing audio segmentation. The goal is to obtain homogeneous acoustic audio segments consisting of events such as music, speech or silence. One also wants to add necessary notes about speaker identity, type of recording and so on. The latter has

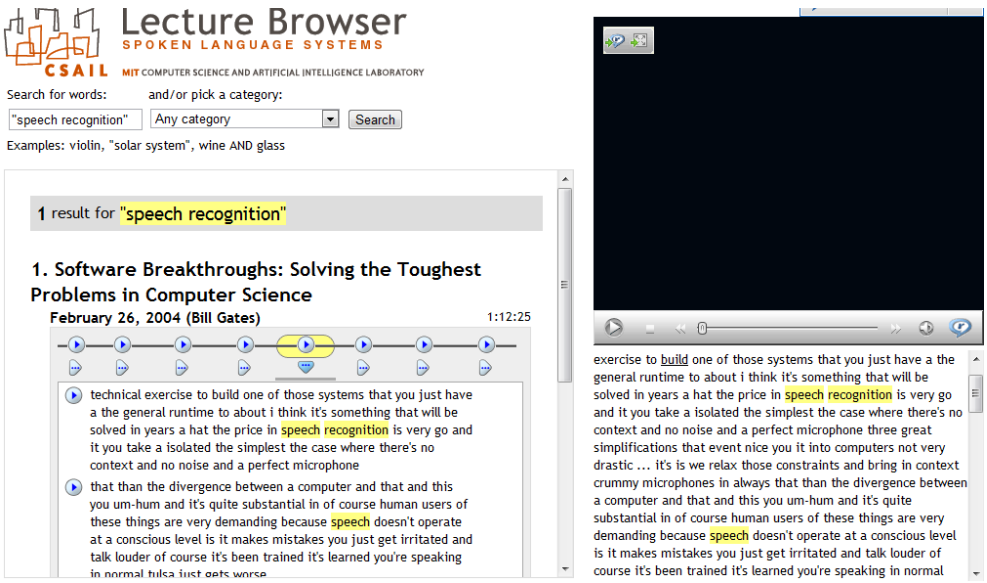


Figure 1.1: The MIT lecture browser.

influence on what content one can expect. In telephone recordings there is more noise than in studio recordings. In these recordings there is also mostly spontaneous speech, which includes more disfluencies such as restarts and hesitations than read text from a script in studio. By detecting the different events one can structure the audio file into parts analogous to pages and paragraphs in text files. There are several challenges in the work of making an event log. Problems include detection of rapid changes, detection of speech in noisy environments, how to handle cases where different kinds of audio occur simultaneously, etc. In news broadcasts, such events are common. For example, new topics are often presented with background music, field reports often contain background sounds such as traffic noise or background speech and debates contain rapid changes between speakers.

Manual transcribing and annotation of audio to learn the content of a file will reduce the information search problem in audio to the problem one has with text search. Yet, this is expensive and it might even be impossible due to privacy concerns [3]. Automatic indexing is therefore desirable. A perfect automatic speech recognition (ASR) system combined with the technology used for text indexing would be the ideal system. Though, today's ASR systems are not robust enough. Another difficulty with this approach is how to handle music and other non-speech sounds. A problem when searching for audio files is to overcome the linearity [4]. When a relevant document is found, one has to listen to the whole file to be sure nothing is missed. This is

more time consuming than just skimming through a text file. Indexing can be helpful for faster approving of the returned file.

This thesis concentrates on two tasks, namely long pause detection and speaker change detection. Much research has already been done in the field and different methods have been compared to each other. However, as corpora have different content varying results may be obtained with the same methods on different corpora. This thesis will use methods presented in others' work, but adapt them to make as good results as possible for the specific corpus used, which contains news broadcasts from NRK (The Norwegian Broadcasting Corporation) radio.

1.2 Work

The first task in this thesis is to segment the news broadcasts into two parts, namely sound and long pauses. This step is meant as a preprocessing for further segmentation such as speaker change detection, where one wants the data to consist of as clean speech as possible. In fact, parts with music or other non-speech sounds would also be desirable to remove before speaker segmentation, but such a task has not been regarded in this master thesis. Another reason to detect long pauses is that they often represent a change of topic or speaker. Pauses shorter than 1 second occur naturally during speech, and are not desirable to detect.

The most common methods that have been used for segmentation tasks are model-based, metric-based and energy-based segmentation techniques. In this thesis, model-based segmentation will be used for pause detection. An advantage with this method is that when the models are trained on example data, one often gets a better basis for succeeding in the decoding than when using a method that does not need training. On the other side, training is a time consuming process and one is dependent on enough representative training data. [5] used hidden Markov models for speech/non-speech separation and [6] used Gaussian mixture models for classifying into wide band speech, pure narrow band speech, music and speech and only music. In this thesis, a Gaussian mixture model will be implemented for each of the two events: sound and long pause. Gaussian mixtures are capable of modeling arbitrary densities [7], which is useful in audio segmentation. The experimental results will be compared to a manual transcription, which has been carried out by students at an earlier stage. The transcription denotes the start and end times of the pauses and gives information about the appurtenant background conditions. Both classification and recognition will be performed. In classification, the recognizer must decide whether defined segments are

sound or long pauses, while in recognition, the recognizer must find the segment boundaries as well. Realistic results must be obtained from recognition, but classification is useful for tuning of parameters and optimization of the method.

The second task is to perform speaker segmentation, i.e. detect points in time where speaker changes occur. As the number of speakers and their identities are unknown, model-based segmentation is not applicable. Another possibility for speaker segmentation is to use an energy-based approach, which assigns changes where periods of silences occur. However, this may not be useful for news broadcasts, as these often contain rapid changes between speakers [8]. A metric-based approach is therefore chosen. In this method, the similarity between to adjacent windows moving over the audio signal is measured. [9] used a distance measure called the Bayesian information criterion (BIC), while [1] presented a modified version of this criterion. Both methods will be implemented in this thesis and different parameter values and options will be tested. [10] used another distance measure called the symmetric Kullback-Leibler (KL2) distance. The KL2 distance will in this thesis be used as a post-processing step with the purpose of reducing the number of false change points introduced by the BIC algorithm.

In order to be able to compare the obtained results with results achieved by others, the algorithms used must have been applied on the same data. Several authors have published results obtained on a corpus called the HUB-4 1997 evaluation data. This corpus will therefore be used to make comparative tests.

1.3 Outline of the report

The report is structured in the following way:

Chapter 2 contains the theory behind parameterization of audio, hidden Markov models, Gaussian mixture models, Bayesian information criterion, symmetric Kullback-Leibler distance and different evaluation measures. This chapter will make a foundation for understanding the rest of the report.

Chapter 3 contains information about the corpora that have been worked on. This includes information about the audio, and the accompanying transcriptions.

Chapter 4, 5 and 6 presents the experiments that have been performed. Chapter 4 gives the detailed approaches to each problem and chapter 5 presents the results of them. Chapter 6 presents the discussion of

the results, in addition to mentioning special difficulties and problems. Chapter 5 also gives a suggestion to further work.

Chapter 7 summarizes the most important results and aspects with this report.

Chapter 2

Theory

This chapter presents the theory behind the tasks that have been performed in this thesis. The different segmentation techniques require a parameterization of the audio. In this process the most important features of the audio are extracted. The features used, mel frequency cepstral coefficients, are explained in section 2.1. The aim of the first task was to detect long pauses. A Gaussian mixture model was fitted to each of the two events, long pause and sound, in order to describe them as good as possible. The models were implemented in the hidden Markov model toolkit as one-state hidden Markov models. The two model types and their connection are presented in section 2.2. In the second task, different metric-based algorithms for speaker segmentation were implemented. These are presented in section 2.3. Finally, the evaluation criteria used are explained in section 2.4.

2.1 Feature extraction

In audio segmentation, the sound wave is not used directly, but it is parameterized beforehand. This parameterization should extract features that are good at distinguishing between the different classes of interest. Features within the same class should be as similar as possible. The desired features will vary for different applications. In speech recognition for example, one wants to recognize the speech, but the speaker identity is irrelevant. In speaker segmentation on the other hand, the goal is to separate the different speakers. One alternative of acoustic features for representing audio is mel frequency cepstral coefficients (MFCC). MFCCs are commonly used in speech recognition, but they have also been used in audio classification due to good discrimination abilities [11]. How to obtain the MFCCs are explained below. The steps are displayed graphically in figure 2.2.

1. Divide the audio signal into overlapping frames.
2. Weigh each frame with a window function.
3. Take the fast Fourier transform (FFT) of the magnitude spectrum of the windowed data.
4. Perform mel-filtering. This is done by using triangular filters uniformly spaced at the mel scale. The mel scale is defined as

$$Mel(f) = 2595 \log_{10}(1 + \frac{f}{700}) \tag{2.1}$$

   where f is the frequency in Hz. Using the mel scale will result in a better model of the hearing. The filter bank is shown in figure 2.1. As seen, the total response of the filter bank equals 1 in all bands except the first and the last.

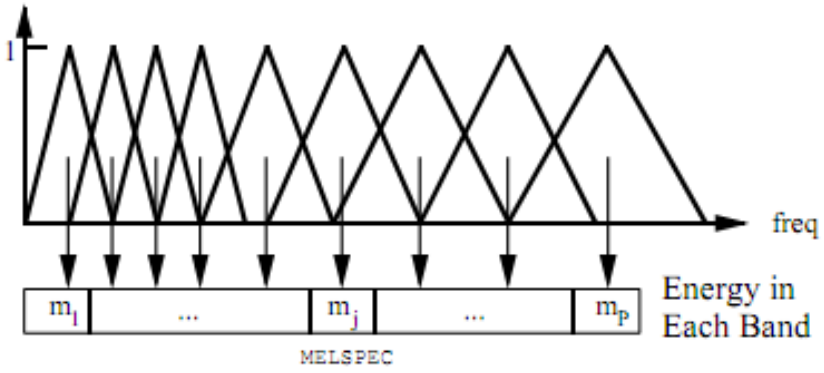


Figure 2.1: Mel-scale filter bank [12].

5. Compute the log-energy at each filter output.
6. Perform a discrete cosine transform (DCT) of the log energies of each filter output. This results in the MFCCs.

It is assumed that the content in each frame is stationary. Speech is not stationary, but the assumption will be approximately true if a short enough window length is chosen. Typically, the window is a Hamming window of 25 ms applied on overlapping frames shifted by 10 ms [14]. A longer shift will give a lower time resolution, and one can miss important information. On the other hand, there will be less need for storing and computation. With a

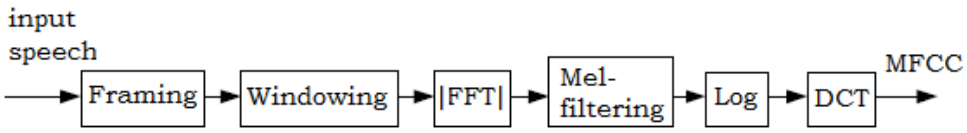


Figure 2.2: Steps for calculating MFCCs [13].

longer window, one risk that the assumption of stationary frame content is not valid. By contrast, a shorter window gives poorer frequency resolution.

A common choice is to calculate 12 MFCCs for each window, with the 0th cepstral parameter, C_0, appended as the energy component. Another possibility is to perform cepstral mean normalization (CMN). Channel variations, like different microphones and room acoustics, may be a problem in recognition and segmentation tasks. CMN of the MFCCs can give increased robustness as it handles convolutional distortions [14]. The CMN can be performed as follows.

Given a set of T cepstral vectors $\mathbf{X} = \{\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_{t-1}\}$. The sample mean $\overline{\mathbf{x}}$ is given by

$$\overline{\mathbf{x}} = \frac{1}{T}\sum_{t=0}^{T-1}\mathbf{x}_t \tag{2.2}$$

The sample mean is then subtracted from each vector $\mathbf{x}_t$, which results in the normalized cepstrum vector $\hat{\mathbf{x}}_t$

$$\hat{\mathbf{x}}_t = \mathbf{x}_t - \overline{\mathbf{x}} \tag{2.3}$$

Denote $y[n]$ as the output of passing $x[n]$ through a filter $h[n]$. In the cepstral domain, the filter $\mathbf{h}$ is given by

$$\mathbf{h} = \mathbf{C}\left(\ln|H(\omega_0)|^2 \cdots \ln|H(\omega_M)|^2\right) \tag{2.4}$$

where $\mathbf{C}$ is the DCT matrix. Since convolution in the time domain corresponds to summation in the cepstral domain one has

$$\mathbf{y}_t = \mathbf{x}_t + \mathbf{h} \tag{2.5}$$

This results in the sample mean $\overline{\mathbf{y}}$

$$\overline{\mathbf{y}} = \frac{1}{T}\sum_{t=0}^{T-1}\mathbf{y}_t = \frac{1}{T}\sum_{t=0}^{T-1}(\mathbf{x}_t + \mathbf{h}) = \overline{\mathbf{x}} + \mathbf{h} \tag{2.6}$$

and the normalized cepstrum is given by

$$\begin{aligned}\hat{\mathbf{y}}_t &= \mathbf{y}_t - \overline{\mathbf{y}} \\ &= \mathbf{x}_t + \mathbf{h} - \overline{\mathbf{x}} - \mathbf{h} \\ &= \hat{\mathbf{x}}_t\end{aligned} \tag{2.7}$$

From equation (2.7) it is clear that the normalized cepstrum is not affected by linear filtering operations. For the CMN to be useful, the content in a specific environment must last longer then 2-4 seconds [14].

Delta coefficients can be used to measure temporal changes in the MFCCs [14, p. 425], and is also often desirable as speech is not stationary. The 1st order delta coefficient at time t can be computed as

$$\Delta c_t = \frac{\sum_{\theta=1}^{\Theta} \theta \left(c_{t+\theta} - c_{t-\theta} \right)}{2 \sum_{\theta=1}^{\Theta} \theta^2} \tag{2.8}$$

where c_t is the static MFCC at time t. A typical value for the window size Θ, is 2. The same formula can be applied on the 1st order delta coefficients to obtain 2nd order delta coefficients.

2.2 Model-based segmentation

Model-based segmentation techniques are often used when the set of events to detect is known. In these methods, a set of models are trained on example data of the different events, prior to the segmentation. In the following section, hidden Markov models (HMM) and Gaussian mixture models (GMM) will be presented. GMMs have been used in this thesis for detection of long pauses. It will be shown that these models can be implemented as HMMs, and thus the hidden Markov model toolkit (HTK) can be used for implementation of the models.

2.2.1 Hidden Markov models

A hidden Markov model (HMM) is a statistical model that can be used to model time varying processes, such as the vector sequences obtained from parameterization of an audio signal. The following description is inspired by [15, p. 8-20], where a more detailed description can be found. A HMM generates a discrete time signal from a series of connected states. An example of a HMM with $(N-2)$ emitting states is shown in figure 2.3.

For each frame or time step, the model changes state according to a set of transition probabilities $\{a_{ij}\}$, which denotes the probability for going from

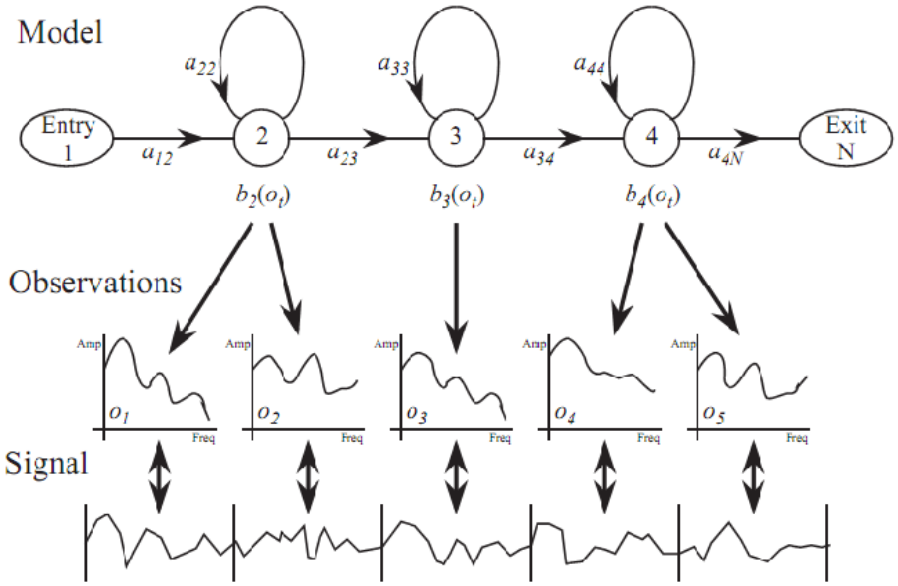


Figure 2.3: Example of a hidden Markov model [15].

state i to state j. This can be mathematically expressed as

$$a_{ij} = P\left(x(t+1) = j | x(t) = i\right) \tag{2.9}$$

where $x(t)$ is the state at time t. The entered state then generates an observation according to the output probability distribution $b_j(\mathbf{o}_t)$. The entry and exit states only denote the start and end of the model and do not result in an output. For continuous output distributions, as is used in this thesis, $b_j(\mathbf{o}_t)$ is the likelihood of state j generating the observation vector $\mathbf{o}_t$. This can be expressed mathematically as

$$b_j(\mathbf{o}_t) = P(\mathbf{o}_t | x(t) = j) \tag{2.10}$$

The output distribution may e.g. be a Gaussian distribution as will be presented in section 2.2.2. The observations are assumed to be independent of each other. Another assumption is that the probability of being in state $x(t)$ only depends on the previous state $x(t-1)$.

The model parameters are estimated by using example observation sequences of a known class. This is called training, and can be done using the Baum-Welch algorithm. The algorithm is complex and will not be described here, but it is presented in [14, p. 389-393]. The models are named hidden Markov models, because it is normally only the signal and model parameters that is known, while the state sequence is hidden. Finding the state sequence

is called The Decoding Problem and can be solved using the Viterbi algorithm. This algorithm chooses the most likely state sequence up to state i for each time t, and remembers it. The best sequence can then be decided by an induction procedure, given as

$$\phi_j(t) = \begin{cases} a_{1j} b_j(\mathbf{o_1}) & \text{if } t = 1;\ 2 \leq j \leq N-1 \\ \underset{2 \leq i \leq N-1}{Max} \Big[\phi_i(t-1) a_{ij}\Big] b_j(\mathbf{o_t}) & \text{if } 2 \leq t \leq T;\ 2 \leq j \leq N-1 \\ \underset{2 \leq i \leq N-1}{Max} \Big[\phi_i(T) a_{iN}\Big] & \text{if } t = T^+;\ j = N \end{cases} \tag{2.11}$$

where $\phi_j(t)$, is the likelihood of the most likely state sequence up to state j at time t, which has generated the observation sequence $\mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_t$. State 1 and state N are non-emitting. The model always starts in state 1 at time 1^-, and ends in state N at time T^+. A back pointer must be saved for each time step, to recover the best state sequence.

$$\chi_j(t) = \begin{cases} \underset{2 \leq i \leq N-1}{Argmax} \Big[\chi_i(t-1) a_{ij}\Big] & \text{if } 2 \leq t \leq T;\ 2 \leq j \leq N-1 \\ \underset{2 \leq i \leq N-1}{Argmax} \Big[\chi_i(T) a_{iN}\Big] & \text{if } t = T^+;\ j = N \end{cases} \tag{2.12}$$

The most likely state sequence $X = x(1), ..., x(T)$ is then recovered as

$$x(t) = \begin{cases} \chi_N(T^+) & \text{if } t = T^+ \\ \chi_{x(t+1)}(t+1) & \text{if } 1 \leq t \leq T-1 \end{cases} \tag{2.13}$$

The likelihoods in the Viterbi algorithm will in the end decrease to a very small number. To avoid underflow when using computers for calculation, the Viterbi algorithm is implemented in the logarithmic (log) domain. An audio segment is acoustically closer to a HMM, the higher log likelihood it has.

2.2.2 Gaussian mixture models

A Gaussian mixture model (GMM) is a classic parametric model often used in pattern recognition techniques [16]. In a GMM, the probability distribution for the observed parameters is given by

$$p(\mathbf{o}) = \sum_{i=1}^{M} \alpha_i \mathcal{N}(\mathbf{o}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_\mathbf{i}) \tag{2.14}$$

where $\mathbf{o} = \{\mathbf{o_t}\}$ are independent observation vectors, α_i are normalized positive scalar weights, M is the number of mixtures and $\mathcal{N}(\mathbf{o}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the p-dimensional normal distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ defined by

$$\mathcal{N}(\mathbf{o}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{p/2} |\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{o} - \boldsymbol{\mu})^t \boldsymbol{\Sigma}^{-1} (\mathbf{o} - \boldsymbol{\mu})\right] \tag{2.15}$$

Figure 2.4 shows the depiction of an M component Gaussian mixture density.

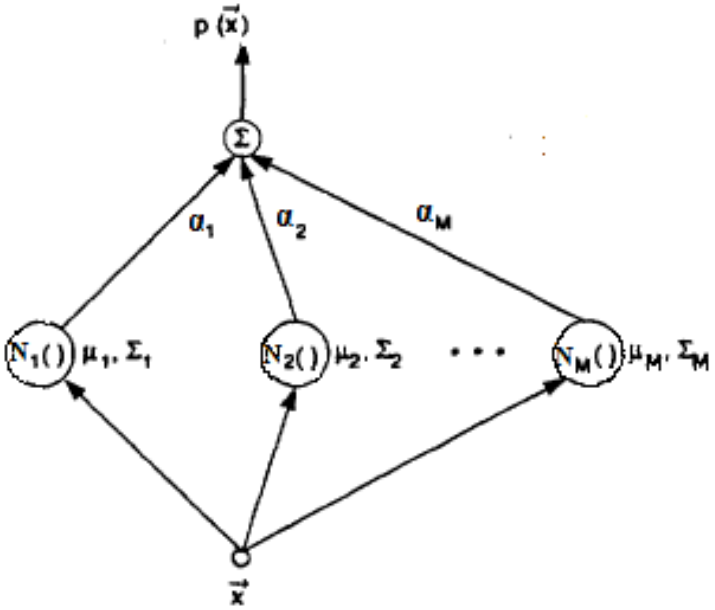


Figure 2.4: Depiction of an M component Gaussian mixture density [7].

The performance of a system depends on the number of chosen mixtures. This number can vary from task to task, and the best solution is to determine it experimentally. Too few mixture components can result in inaccurate models. Larger corpuses of training data therefore normally need more mixtures in order to get accurate models. At the same time, choosing too many mixtures can give reduced performance if the number of model parameters becomes large relative to the available training data [7]. The computational cost increases with an increasing number of mixtures. Figure 2.5 is a plot from MATLAB of a bivariate Gaussian distribution with two mixture components. The plot has two peaks because of the two mixture components.

The GMM can be thought of as a hidden Markov model with only one emitting state and with the output probability distribution, $b_j(\mathbf{o}_t)$, given as

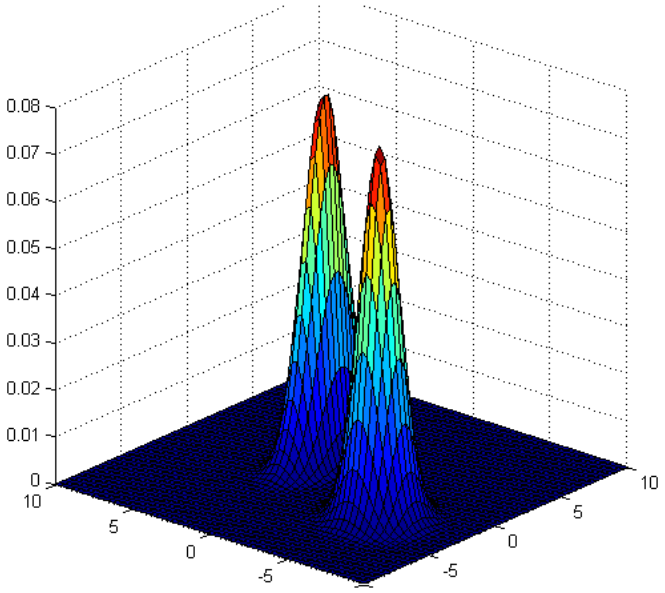


Figure 2.5: Plot of a bivariate Gaussian distribution with two mixture components.

the Gaussian distribution in equation (2.14). Thus, the same methods for training and testing can be used for the GMMs as for HMMs.

2.3 Metric-based segmentation

Metric-based segmentation techniques are often used in change detection tasks. In these methods, two neighboring windows are moved over the audio signal, and a similarity measure is used to compute the resemblance between the two windows. Different similarity measures can be used, and some of them have been investigated in this thesis. For the initial change point detection the Bayesian information criterion (BIC) and a modified version of this measure presented in [1], were tested. For reduction of false alarms, a method called the symmetric Kullback-Leibler (KL2) divergence, presented in [8], was used. The methods will be explained in the following sections.

2.3.1 Bayesian information criterion

In BIC, two relatively small adjacent windows are moved over the audio signal, as shown in figure 2.6.

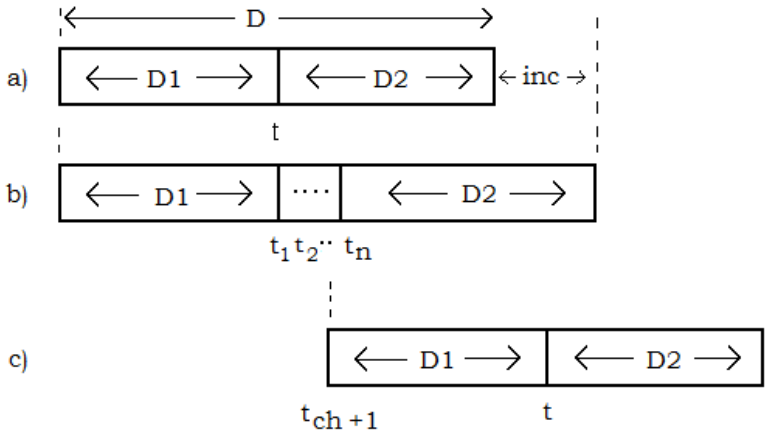


Figure 2.6: a) Two neighboring windows with data $D1$ and $D2$ around time t, where a distance measure is used to find out if there exists a change point or not [1]. b) If no change point is found, the window size is increased with an amount *inc*. The distance measure in calculated at each time instance t that assures a minimum length of the two windows. c) If a change point, t_{ch}, is found, the window is shifted to start next to this change point.

The similarity between the two windows with data $D1$ and $D2$ is measured to decide if there is a change point at time t or not. BIC uses hypothesis testing for this. It is assumed that the data can be modeled by Gaussian probability density functions (PDF). The two hypothesizes are

H_0: A change point occur at time t. The data sets $D1$ and $D2$ are modeled by two Gaussian mixture models.

H_1: There is no change point at time t. The data sets $D1$ and $D2$ are modeled by a single Gaussian mixture model.

In the case of H_0, the data in the adjacent windows are believed to come from different sources. Thus, the two data sets should be modeled by two individual Gaussian mixture models with parameters θ_1 and θ_2 respectively. In the case of H_1 the data in the adjacent windows are believed to come from the same source, and the data sets should be modeled by one Gaussian mixture model with parameters θ. For each time t, the LL of the data in both hypotheses are compared. The log likelihood (LL) of the observation vectors $\mathbf{o}$ is given by

$$LL(\mathbf{o}) = \log p_n(\mathbf{o}|\theta) = \sum_{k=1}^{n} \log p(\mathbf{o}_k|\theta) \tag{2.16}$$

where $p(\mathbf{o})$ is the multivariate Gaussian PDF with M mixture components described in equation (2.14) and n is the number of observation vectors in $\mathbf{o}$. The BIC is defined as

$$BIC(t) = LL(D_1|\theta_1) + LL(D_2|\theta_2) - LL(D|\theta) - \lambda\frac{\Delta K}{2}\log N \qquad (2.17)$$

where λ is a penalty factor, ΔK is the difference in the number of parameters used in the two hypothesizes and N is the numbers of data points in window D. If $BIC(t) > 0$, a change point is considered.

In the original version of BIC, one mixture component is used for each model. In the modified version presented in [1], the data sets in the case of H_1 are modeled by a GMM with two mixture components. This model can be obtained by using the Expectation-Maximization (EM) algorithm, described in [14, p. 170-172]. Using a two mixture model in the case of H_1 and single mixture models in the case of H_0, gives $\Delta K = 0$ in equation (2.17). The last term will therefore be eliminated and the decision of whether there is a change or not now depends on the differences in likelihoods given by

$$\Delta L = \log L(D_1|\theta_1) + \log L(D_2|\theta_2) - \log L(D|\theta) \qquad (2.18)$$

If $\Delta L > 0$, a change point is considered.

Parameters that must be chosen are minimum window size, window increase and maximum window size. In addition one can choose to define a change point for the first time instance t where $\Delta L > 0$, or find all time instances within a window where $\Delta L > 0$ and select the one with the highest value. The algorithm for both BIC and the modified version goes as follows.

```
1. initialize the window D ∈ [a,b] with a = 1 and
   b = minimum window size of D (in frames).
2. measure the BIC value for each t in the window that
   assures a minimum length, min_window, of D1 and D2.
3. if(no change in [a+min_window,b-min_window])
     b = b + frame_increment;
   else(t is the change point)
     a = t + 1, b = a + min_window*2 - 1;
4. if (b-a > maximum window size)
     a = b - max_window;
5. go to (2)
```

2.3.2 Symmetric Kullback-Leibler

The symmetric Kullback-Leibler (KL2) distance, is another distance measure between data in two neighboring windows. A segmentation algorithm based

on this measure is presented in [8]. In this thesis, the measure has been used in a post-processing algorithm for trying to reduce the number of false alarms.

The KL2 distance between two neighboring windows having feature vector sequences $D1$ and $D2$ is given by

$$\mathrm{KL2}(D1, D2) = \int_o \left[p_1(\mathbf{o}) - p_2(\mathbf{o})\right] \log \frac{p_1(\mathbf{o})}{p_2(\mathbf{o})} \, \mathrm{d}\mathbf{o} \tag{2.19}$$

If assuming that $D1$ and $D2$ can be modeled by a multivariate Gaussian distribution, such that $p_1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $p_2 \sim \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$, where $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ was defined in equation (2.15), the KL2 distance can be written as

$$\begin{aligned} \mathrm{KL2}(D1, D2) = \; & \tfrac{1}{2}\mathrm{Tr}\left[(\boldsymbol{\Sigma}_1 - \boldsymbol{\Sigma}_2)\left(\boldsymbol{\Sigma}_2^{-1} - \boldsymbol{\Sigma}_1^{-1}\right)\right] \\ & + \tfrac{1}{2}\mathrm{Tr}\left[\left(\boldsymbol{\Sigma}_1^{-1} - \boldsymbol{\Sigma}_2^{-1}\right)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T\right] \end{aligned} \tag{2.20}$$

A change point is accepted if the KL2 distance exceeds a given threshold. This threshold can be set automatically, as the mean of KL2 distances measured at all time steps t, shifted with an amount ls, in a distance $\pm Tmax$ around the proposed change point t_{ch}:

$$\mathrm{threshold}_{ch} = \alpha \cdot \frac{1}{2T_{max} + 1} \sum_i \mathrm{KL2}_{ch+i} \tag{2.21}$$

where $-\mathrm{T_{max}}/\mathrm{l_s} < i < \mathrm{T_{max}}/\mathrm{l_s}$, and α is a pre-defined scaling factor that must be set experimentally.

2.4 Evaluation criteria

In evaluation of segmentation experiments, one often divides the result into four groups, depending on the result of the classification. Say one want to separate an event X, from other events Y. The four possible results of the segmentation are:

- Correct acceptances (CA): number of segments classified as X and belonging to X
- False alarms (FA): number of segments classified as X, but belonging to Y
- Correct rejections (CR): number of segments classified as Y and belonging to Y

- False rejections (FR): number of segments classified as Y, but belonging to X

It is often easier to evaluate the result if it is given in percents or rates. The two error types can be presented as false alarm rate (FAR) or false rejection rate (FRR)

$$\mathrm{FRR} = \frac{\mathrm{FR}}{\mathrm{FR} + \mathrm{CA}} \tag{2.22}$$

$$\mathrm{FAR} = \frac{\mathrm{FA}}{\mathrm{FA} + \mathrm{CR}} \tag{2.23}$$

Other evaluation scores commonly used are precision, recall and F-score. In a general case, these can be defined as

$$\mathrm{Precision} = \frac{\text{number of correctly returned results}}{\text{number of alleged correct results}} \tag{2.24}$$

$$\mathrm{Recall} = \frac{\text{number of correctly returned results}}{\text{number of possible correct results}} \tag{2.25}$$

$$\text{F-score} = 2 \cdot \frac{\mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}} \tag{2.26}$$

Chapter 3

Databases

This chapter describes the two databases used in this thesis. RUNDKAST, described in section 3.1, has been used in experiments in order to adapt the algorithms to give optimal performance on this corpus. The HUB-4 1997 evaluation data, presented in section 3.2, has been used to compare the results obtained in this thesis with results obtained by others.

3.1 RUNDKAST

The data used for experiments in this thesis, is from a corpus containing news broadcasts from the Norwegian Broadcasting Corporation (NRK). NRK is Norway's major broadcasting institution, with several TV channels as well as radio channels. The corpus consists of different radio news programs with varying content: read texts, spontaneous speech, dialogs, music, speech in various Norwegian dialects and speech in other languages. There are also variations of speakers, recording type and quality, background noises, etc. Programs with focus on sports, culture, or economics were excluded from the corpus. NRK does not have commercials. The corpus contains about 77 hours of audio and it was produced by the RUNDKAST project [17] at the Norwegian University of Science and Technology (NTNU). The audio files have a sampling frequency of 16000 Hz.

Each audio file has an accompanying transcription, containing manually segmented, labeled[1] and transcribed[2] audio. The transcription files were given in a XML-based format defined by version 1.5.1 of the Transcriber tool [18]. Each audio file is split into 3 hierarchical annotation levels.

[1] adding notes to different events

[2] speech sounds represented as text

1. Sections: registration of whether the file part is a report (regular news items), filler (headlines, small talk etc) or a portion not to be transcribed (music, jingles, speech not relevant for the program, etc.)
2. Speaker turns: registration of points in time for speaker changes and identification of the speakers. For each speaker turn, the speaker mode (spontaneous or planned) and the type and fidelity of the recording are also denoted. The recording type can either be studio recording or telephone recording, and the fidelity can be high, medium or low.
3. Segments: defining smaller parts of the file, to make the transcription easier. There is a new segment for every section, every change of speaker and every lasting change in background conditions. There is also often a new segment at breathing pauses. The segments should ideally have duration of 2-5 seconds. Pauses of one second or longer were also asked to be separated out as segments and these segments will naturally be shorter.

For each level the start- and end times are given. In addition there is a level for lasting background conditions. The background boundaries must coincide with segment boundaries, but may span several segments, speaker turns or sections. The transcription files also include labeling of different events. These events can be music, pauses, inhalations, background noise of short duration, overlapping speech etc.

Figure 3.1 displays a screen shot of the transcription tool Transcriber, where the hierarchical structure is shown. From the bottom, the time boundaries of the chosen segment and the time axis are displayed. Next the segments are listed, then the speakers and the sections. Closest to the signal are the background conditions, if any, displayed. The four levels are assigned with different colors.

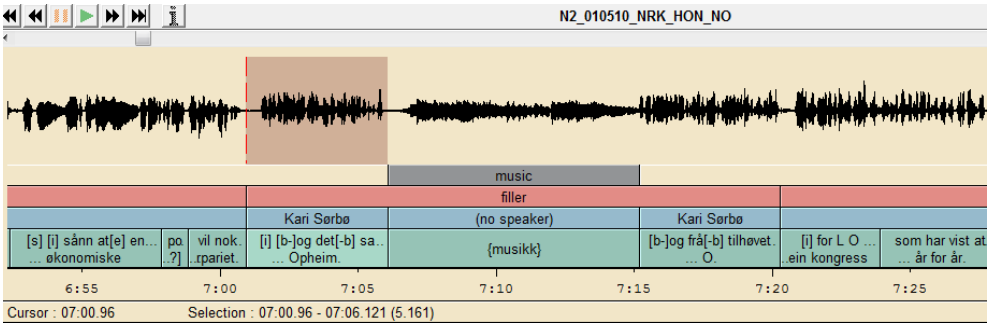


Figure 3.1: Screen shot of the transcription tool Transcriber [18].

3.2 HUB-4 1997

The HUB-4 1997 evaluation data is administered by the NIST Spoken Natural Language Processing Group [19]. The database consists of nearly 3 hours of concatenated radio and television broadcast news stories. The news contains variations similar to those found in RUNDKAST; variations of speakers, different quality of recordings, different backgrounds conditions, planned or spontaneous speech, etc. Commercials and sports results are excluded in the transcription as the syntax and semantics differ from the rest of the data. The data is monophonic with a sampling frequency of 16000 Hz.

The accompanying transcription contains information similar to what is found in RUNDKAST. This includes division into sections, speaker changes and segments, and labeling of speaker mode, fidelity and significant background conditions. Other information that is included in the transcription is overlapping speech, non-speech sounds such as mispronunciations, inhalations and coughing, and segments not to be transcribed (including the reason, i.e. foreign language, commercials, etc.).

Chapter 4

Method

This chapter describes the approaches to the different experiments performed. Section 4.1 describes the pause detection, including details around the feature extraction, the training and testing of Gaussian mixture models and the evaluation criterions used. Section 4.2 describes the speaker segmentation, including details around the feature extraction, the different experiments performed and how the results were evaluated.

4.1 Pause detection

4.1.1 Feature extraction

MFCCs were used as feature vectors for pause detection. The vectors were extracted with the function `HCopy` from HTK. Detailed information about this toolkit and its functions can be found in [12]. 12 static MFCCs were used with the 0th coefficient appended as the energy coefficient. In some experiments, delta coefficients were also included. CMN was performed in order to try to reduce the influence of channel noise. Experiments were done with different window lengths and window shifts. The combinations 30/15 ms and 40/20 ms for length/shift were tested.

4.1.2 Implementation of methods and experiments

The aim of the first task was to separate the audio into two parts; sound and long pauses. A long pause has been defined as a period of silence or background noise (grating or car noise), preferably with a duration of 1 second or longer. Filled pauses, i.e. pauses containing background speech, laughter, coughing or similar, were considered as sound. The pause detection was performed by modeling each of the two events with a GMM. The GMMs

were implemented in HTK as HMMs with one emitting state. The functions `HInit` and `HRest` were used for training, and `HVite` and `HResult` were used for testing. The training set was taken from RUNDKAST and consisted of about five hours of audio. Of this were almost 10 minutes or 332 segments, long pauses. The pauses were chosen from segments that were labeled as long pause by the transcribers and that contained silence or background noise. A few segments had unknown content as they were not transcribed and labeled with background level 'off'. These were listened to and re-labeled as long pauses if the conditions were met. The different segments chosen for training were picked by creating virtual files in HTK [12, p.51], which linked to different segments in the real file. The data used for training and the data used for testing were chosen such that they contained material from different news programs and from different parts of each program. The training and test sets did not consist of data from the same files. Figure 4.1 shows a simple overview of the system.

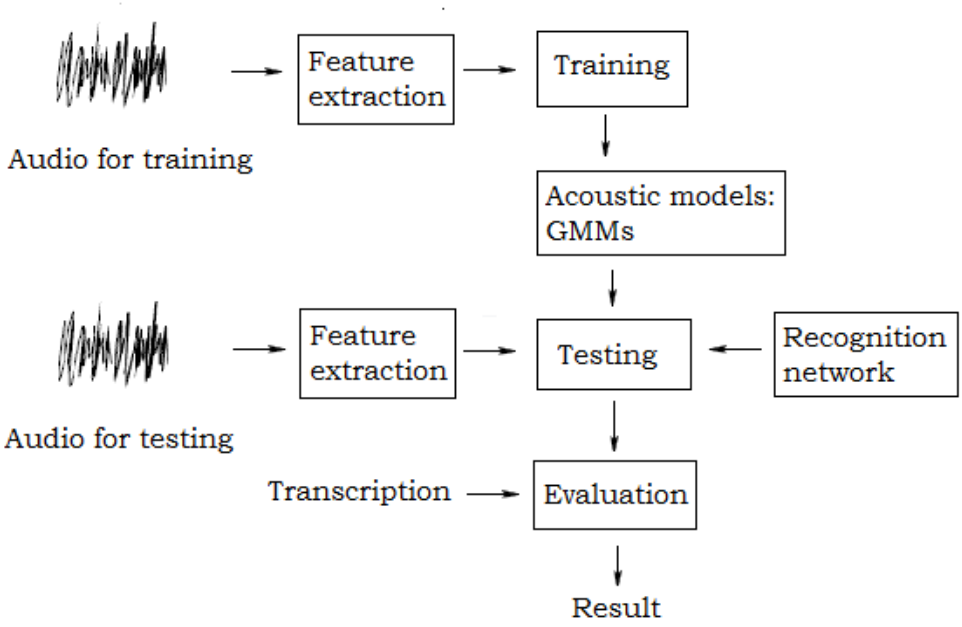


Figure 4.1: Model-based segmentation system using GMMs.

Classification was performed to investigate how well the GMMs could separate sound and pauses when the segment boundaries[1] were known. The

[1]Point in time where one segment ends and another starts. A segment is here a continuous part of the audio that is labeled as either long pause or sound

testing was carried out by making virtual files of all segments that should be classified. The chosen segments coincided with the segments defined in the transcription. The recognition network was specified to choose either the label sound or the label long pause for each segment. The test set for classification included 84 segments of long pauses and 1190 segments of sound, chosen from RUNDKAST. Classification experiments were also used to find the optimum audio parameterization and the ideal number of mixture components in the GMMs.

To get realistic results, recognition had to be performed. In this task the system must detect both the correct events and the correct boundaries. Experiments were performed with different values of an insertion penalty, which was set with the option -p in `HVite`. This option adds a penalty to the calculated LL score, for each transition from one model to another. Consequently the number of inserted pauses will be reduced. After the recognition with HVite, segments of 0.9 seconds or shorter were removed from the results, as these segments are too short to be in the target group. The test set for recognition consisted of nearly 18 hours of continuous audio from 26 different files in RUNDKAST. The set included 100 long pauses.

The models were also tested with the HUB-4 1997 data. A new labeling of long pauses in this database was done to use in the evaluation, as the original transcription contained many flaws. The pauses were found by looking for segments with low or stable energy in the audio signal. Only pauses with duration of 1 second or longer were labeled.

An investigation of the performance of the acoustic models was done by running two rounds with recognition; one with only the GMM for pause and one with only the GMM for sound. The recognition network was made so that the probability for leaving the emitting state was 1, and the probability for going from the end state to the start state was 1. In this way one gets the LL scores for both models for each frame. The results were plotted against the frame number, to investigate if it was really possible to decide between sound and pause. The model that fits the data best should have the highest LL score. An ideal plot for the case of a pause between two segments of sound is shown in figure 4.2. The scores for the two models are plotted in different colors. One will not obtain an equally good plot in reality, but with applicable models it should be possible to separate the two graphs.

As an attempt to improve the results, a new training set was chosen. In this set, it was tried to exclude sound segments that contained longer periods of silence. This was done by excluding the start and end of each segment, where silence often exists. In addition, segments with a short pause followed by another non-speech element like inhalation or exhalation were excluded. Only pause segments longer than 0.95 seconds were included in this set. The

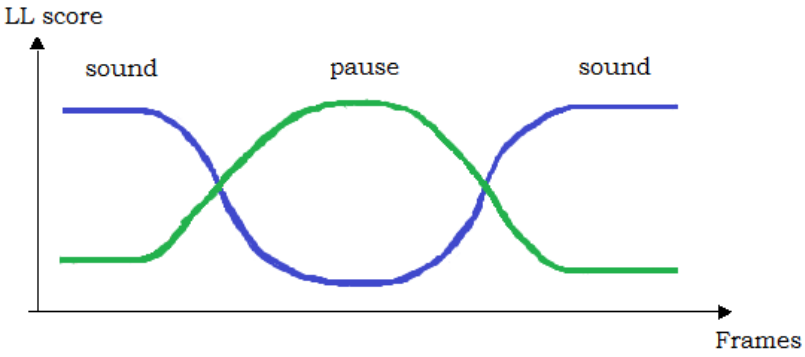


Figure 4.2: The ideal scenario when the LL scores per frame are plotted for the two models. The green plot shows the scores for the pause model, and the blue plot shows the scores for the sound model.

training set included 400 pause segments and almost 6.5 hours of sound. A better routine for choosing the training set made more pauses available than in the last set.

4.1.3 Evaluation

The numbers of false alarms and false rejections were used as evaluation measures for pause detection. Using the definitions in section 2.4, class X gives the long pauses and class Y gives the sound segments. F-score, precision (PRC) and recall (RCL) were also used. The F-score was defined in equation (2.26), while the precision and recall in this case can be defined as in equations (4.1) and (4.2) respectively.

$$\text{PRC} = \frac{\text{\# of segments correctly labeled as long pause}}{\text{number of segments labeled as long pause}} \tag{4.1}$$

$$\text{RCL} = \frac{\text{\# of segments correctly labeled as long pause}}{\text{total number of long pauses}} \tag{4.2}$$

A proposed pause has been accepted as correct if the mismatch from the reference transcription was less than 0.7 seconds.

4.2 Speaker segmentation

4.2.1 Feature extraction

MFCCs were used as feature vectors in the speaker segmentation as well. The features were extracted with `HCopy` in HTK. In experiments with static MFCCs, 24 coefficients were used, as chosen by [1]. In experiments with 1st and 2nd order delta coefficients, 12 static coefficients were used in addition to 12 coefficients for each of the delta orders. The window size of the delta windows, denoted as Θ in equation (2.8), are by default 2 in HTK and these values were kept. CMN was not performed, as changes in the environment often happen simultaneously with speaker changes. For example, people may talk with different distance to the microphone. Further, the energy coefficient was not added, since the energy may vary within one speaker segment. This is in contrast to the pause detection, where energy plays an important role. In all experiments, 10 ms window shift and 30 ms window size were used.

4.2.2 Implementation of algorithms and experiments

The algorithms for both BIC and the modified version of BIC were implemented in MATLAB. The scripts can be found in appendix D.1. Several experiments were performed with variations of the window increase and the maximum and minimum window lengths. The difference in the performance between using full or diagonal covariance matrices and between using feature vectors with or without delta coefficients were also investigated. In addition, different thresholds used for deciding whether there is a change point or not at a given time were tested. A list over all the experiments performed can be found in appendix B. The experiment that gave the best result was also tested with the HUB-4 1997 data and compared with the results obtained by [1] with the same data. A MATLAB function from Voicebox [20], called `gaussmix`, was used for training the two mixture Gaussian model in the modified version of BIC. This function uses the EM-algorithm for training. K-harmonic means with K randomly selected data points was used for initialization of the EM-algorithm. Information about K-harmonic means can be found in [21].

A false alarm compensation (FAC) was implemented to try to reduce the number of incorrectly detected speaker changes. The symmetric Kullback-Leibler (KL2) distance was measured between the data $D1$ and $D2$ centered around each proposed change point, t_{ch}, returned from the initial speaker segmentation. This is shown in figure 4.3. As pointed out by [8], the model of a data set will contain data from two speakers if a change point is missed

in the first place. Therefore a maximum window length, $Tmax$ was used as a restriction. $Tmax$ is chosen as the smallest value of a given length Tm and the distance to the next change point.

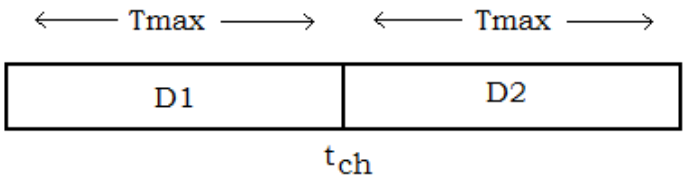


Figure 4.3: Two neighboring windows of length $Tmax$ around a proposed change point t_{ch}. The KL2 distance is used to accept the change point or not.

The threshold was set as in equation (2.21). Figure 4.4 displays graphically the process of finding the threshold. The MATLAB script for the FAC can be found in appendix D.3.

An additional experiment was performed to investigate the effect of including pitch as an acoustic feature. Pitch is a perceptual property of sound, which is closely related to the fundamental frequency, $f0$. A high fundamental frequency makes us perceive a higher pitch. Pitch is also affected by intensity, but intensity is already included in the feature set by using the mel-scale in the calculation of MFCCs. As $f0$ varies from person to person, pitch may be a useful feature in speaker change detection. The pitch values were extracted with a function called `pda` from Edinburgh Speech Tools Library [22]. A description of this algorithm can be found in [23]. The maximum and minimum values of $f0$ were set with the options `-fmin` and `-fmax`. Experiments were done with different values of `-fmin`/`-fmin`. The combinations 50/600 Hz, 60/500 Hz and 70/400 Hz were tested. Low-pass filtering was performed with the option `-L`. The window length and window shift were set with the options `-shift` and `-length` to the same values that were used to extract the MFCCs. Then, one pitch value per MFCC vector was obtained. A pitch detection algorithm from Praat[2] was also considered. A description of this algorithm can be found in [24]. The only option set, was the maximum value of $f0$, and it was kept as the default value of 600 Hz. The minimum value had to be set to 100 Hz in order to get one pitch value per MFCC vector. This is because the window length used for extraction depended on

[2]www.praat.org

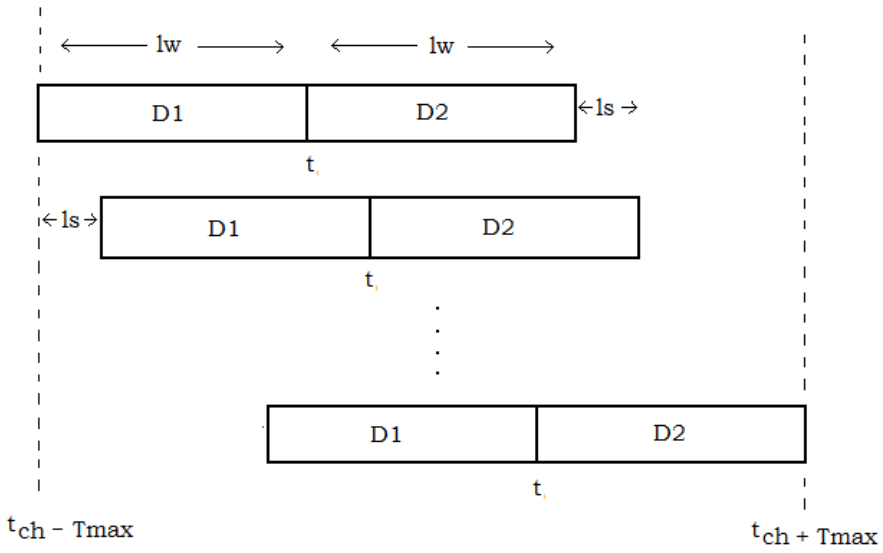


Figure 4.4: Routine for setting a threshold automatically. KL2 distances between to neighboring windows of length l_w around time steps t are measured. The windows are shifted with length l_s, and the threshold is set as a factor multiplied with the mean of all t-values in the range $t_{ch} \pm Tmax$.

the minimum value of $f0$. As a male voice can have an $f0$ as low as 60 Hz [14], it may be difficult to get pitch values from some male voices with this algorithm. The pitch values were modeled with a Gaussian distribution, and the LL scores were calculated from equation 2.16. The LL scores from the pitch values were combined with the LL score from the MFCCs according to the formula

$$\mathrm{LL}_{tot} = \alpha\,\mathrm{LL}_{mfcc} + (1-\alpha)\,\mathrm{LL}_{pitch} \tag{4.3}$$

where the parameter α was set by experimenting. Since unvoiced sounds have no periodicity, they have no defined pitch values. This resulted in less data in the analysis windows with pitch values than in the windows with MFCCs. This again made the LL scores for the pitch values less negative than the LL scores for the MFCCs. To compensate for this, a normalization of the likelihood for pitch was performed according to

$$\mathrm{LL}_{pitch\_norm} = \mathrm{LL}_{pitch}\frac{\mathrm{length}(D_{mfcc})}{\mathrm{length}(D_{pitch})} \tag{4.4}$$

where D_{mfcc} is the analysis window for MFCCs, D_{pitch} is the analysis window for pitch values and LL_{pitch} is the original LL score obtained. Because of the short analysis windows needed for speaker change detection, the windows occasionally contained no defined pitch values. In those cases, only the LL score obtained with MFCCs were used. The code added or changed in the original algorithm for doing this experiment, can be found in appendix D.2.

Differences in the running time of the algorithms between using full covariance matrices and diagonal covariance matrices were measured with the MATLAB functions `tic` and `toc`. In order to get comparable values of the experiments, no other programs were run in the background. The mean of thousand experiments were taken with the maximum and minimum values removed.

The selected test set from RUNDKAST consisted of about 2.75 hours of audio, taken from 11 different files. File parts with a limited number of longer pauses and music segments were evaluated. However, most files included music segments, especially between different topics. Segments with different kind of noise were also present in the test set. How such non-speech segments were treated is explained in the following section. A total of 587 changes were considered after the modifications explained in the following section were performed.

4.2.3 Evaluation

Precision, recall and F-score, defined in equations (2.24), (2.25) and (2.26) respectively, were used for evaluation. More specific for this task, precision

and recall can be defined as in equations (4.5) and (4.6) respectively.

$$PRC = \frac{\text{Number of correctly found change points}}{\text{Number of change points found}} \tag{4.5}$$

$$RCL = \frac{\text{Number of correctly found change points}}{\text{Number of true change points}} \tag{4.6}$$

Only change points with a mismatch with respect to the true change point of less than 1.5 seconds were considered as correct. The test set from RUNDKAST consisted of audio from several different files. The presented results on this test set will be given as the average of the precisions, recalls and F-scores obtained on each file. The HUB-4 data was already concatenated into one file, so the scores presented on this data has been obtained from the whole file.

How to evaluate non-speech events and other irregularities can be discussed. The chosen approach in this thesis is listed below.

Several people talking When several people were talking at once, it was defined to be a speaker change point when the people started talking simultaneously, and a change point when this ended. In cases where it was only a very short[3] contribution by another person such as 'yes', 'mm' or 'but', the duet was ignored.

Music, noise, etc. Segments containing pure music, noise or other non-human sounds, often contributed to a lot of false alarms. Proposed change points within these segments were not included in the evaluation. The change points from speech to non-speech and from non-speech to speech, however, did contribute. Non-speech sounds made by humans, as laughing or coughing were treated as it if they were speech.

Speech in other languages Speech in other languages than Norwegian has not been manually transcribed. Sometimes there was music, several people talking or other interruptions in these segments. They were therefore listened to, and additional change points were marked if necessary.

Pauses Pauses or inhalations sounds were ignored when the duration was around 1 second or shorter.

[3]typically less than 0.5 sec

Chapter 5

Results

This chapter includes the results from the conducted experiments. For the related discussion, see chapter 6. The results from the pause detection are given in section 5.1. This includes the classification results, the recognition results on both RUNDKAST and HUB-4 1997 and the evaluation of the GMMs. In section 5.2, the results from the speaker segmentation are given. This includes results obtained with both BIC and the modified version of BIC. The section also presents a comparison with results obtained by [1].

5.1 Pause detection

5.1.1 Classification

Table 5.1 presents the results from the classification of sound and long pauses, with varying parameterizations and numbers of mixture components. For each parameterization, only the best combination of mixtures is presented. Note that in some experiments, several mixture combinations gave the same results. Table 5.1 displays the combination with the lowest mixture numbers. The complete table can be found in appendix C.

In the rest of the experiments, 1st and 2nd order delta MFCCs, 40 ms window length, 20 ms frame shift, 32 mixtures in the sound model and 16 mixtures in the pause model were chosen. Appendix A.1 displays the configuration file used. Table 5.2 shows the results obtained with the chosen parameters in percents.

5.1.2 Recognition

Table 5.3 displays the results from the recognition of long pauses with the test set from RUNDKAST. An insertion penalty of -145 was chosen. The

Table 5.1: Evaluation of the sound/pause classification for different numbers of mixtures components, dynamic parameters, window lengths and frame shifts. *WL* is window length, *FS* is frame shift. *D* is 1st order delta coefficients, *A* is 2nd order delta coefficients. The notation *x/y mix* means that x mixtures were used for sound and y mixtures were used for pause.

Experiment	FA	FR
D and A, WL = 40ms, FS = 20ms		
32/16 mix	1	1
16/16 mix	2	0
Only D, WL = 40ms, FS = 20ms		
64/32 mix	1	1
D and A, WL = 30ms, FS = 15 ms		
16/16 mix	2	0
Static MFCCs, WL = 40ms, FS = 20		
4/32 mix	1	4

Table 5.2: Results of the sound/pause classification when using the chosen parameters

Experiment	FAR	FRR	F-score
D and A, WL = 40ms, FS = 20ms, 32/16 mix	0.1 %	1.2%	98.8 %

mismatch denotes the difference between the recognized segment boundaries and the segment boundaries marked in the transcription. It is given in milliseconds as differences in start/end boundaries.

Table 5.3: Evaluation of the pause recognition with an insertion penalty of -145.

CA	FA	FR	F-score	Mismatch
45	74	72	38.1 %	109/100 ms

A further investigation of the results was done by listening to the false alarms. It turned out that many of the false alarms contained silence or noise and therefore met the criterion for long pauses. Despite that, these segments had not been marked as long pauses in the reference transcription. Reducing the insertion penalty to -40 resulted in a big increase of false alarms. This is shown in table 5.4. The modified transcription is obtained after all the false alarms were listened to and re-classified as either correct or false alarm. Recall and F-score are not shown for this test as the number of missed pauses was unknown.

Table 5.4: Evaluation of the pause recognition with both the transcription from RUNDKAST and the modified transcription. An insertion penalty of -40 was used.

	CA	FA	FR	PRC	RCL	F-score
RUNDKAST	85	428	32	16.6 %	72.6 %	27.0 %
Modified	478	41	32	92.1 %	-	-

The models were also tested with the HUB-4 1997 data to investigate if it was possible to get better results on another corpus. However, it turned out that also the transcription in this corpus was not accurate with respect to pauses. This corpus seemed to contain fewer long pauses than the test set used from RUNDKAST, so it was a manageable task to go through the data and label all pauses. The results are displayed in table 5.5, showing the scores when the HUB-4 1997 transcription was used and when the modified transcription was used. The results can only be a indication of what is possible to obtain, as the labeling not was carefully done and pauses may have been missed. An insertion penalty of -20 was used. Note that these

results include pauses longer than 1 second, while in the other experiments pauses longer than 0.9 seconds were included.

Table 5.5: Evaluation of the pause recognition on HUB-4 1997 using an insertion penalty of -20. The results are evaluated with both the transcription from HUB-4 and the modified transcription.

	CA	FA	FR	PRC	RCL	F-score	Mismatch
HUB-4	12	108	3	10.0%	80.0 %	17.3 %	292/210
Modified	103	19	27	84.4%	79.2 %	81.7 %	102/79

As the results from the recognition were poor compared to the results from the classification, an investigation of the performance of the acoustics models was done. Figure 5.1 shows the result from this evaluation. The LL scores per frame obtain from each of the two models are plotted.

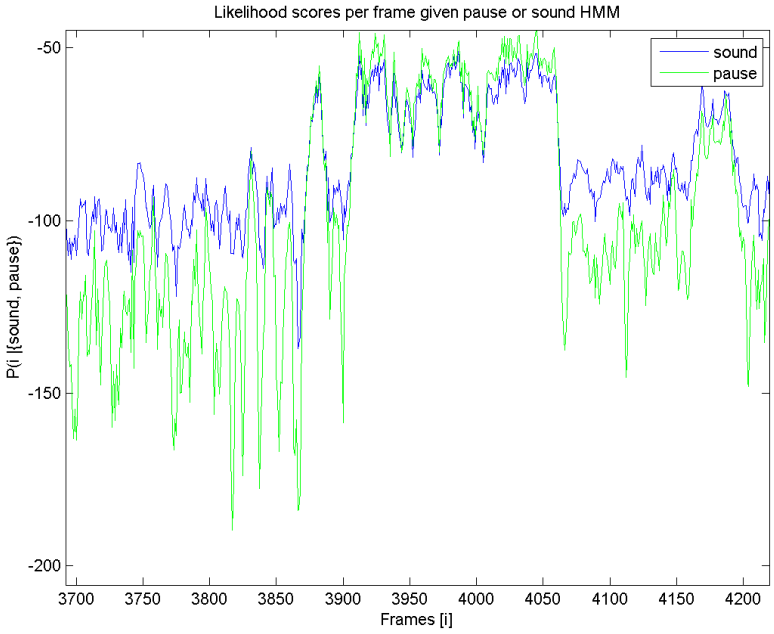
The results obtained with the models trained on the new data set, are displayed in table 5.6. An insertion penalty of -195 had to be used to get comparable results to the first training set. The same mixture combination was used. As the new set contained more data, the mixture combination 64/32 for the sound/pause models was also tested. However, there was not any significant change from the displayed results.

Table 5.6: Evaluation of the pause recognition with models from a new training set. An insertion penalty of -195 was used.

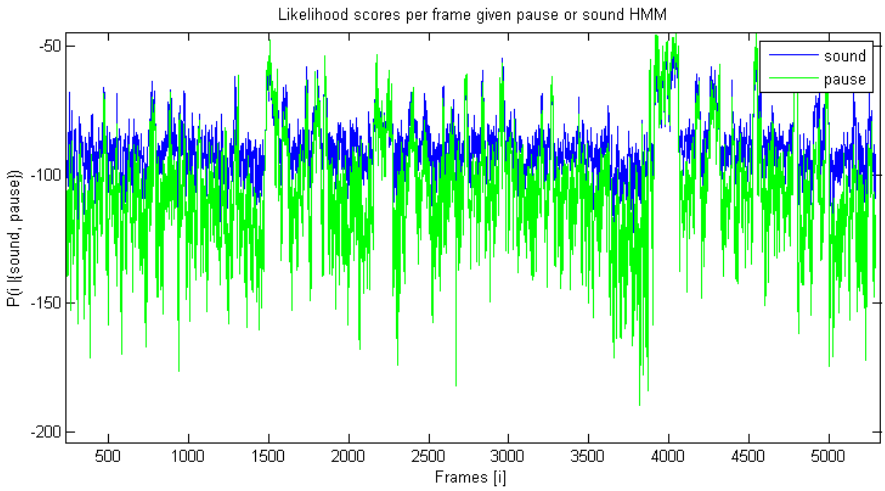
	CA	FA	FR	F-score	Mismatch
New GMMs	45	78	72	37.5 %	117/88

5.2 Speaker segmentation

The best results from the speaker segmentation performed on RUNDKAST are displayed in table 5.7. These were obtained with the modified version of BIC using a threshold of zero, a minimum window of 2 seconds, a maximum window of 8 seconds and a window increment of 1 second. The change points were chosen as the points in time with the highest BIC score larger than zero within a window. The thresholds in the FAC algorithm were multiplied with



(a) Part of a audio file. There is a pause from about frame 3905 to frame 4065.



(b) Part of a audio file. There are three pauses with centers around frame 1520, 2220 and 3980.

Figure 5.1: LL scores per frame for both the sound and pause model. Green graph shows the scores using the pause GMM, and the blue graph shows the scores when using the sound GMM.

$\alpha = 0.45$. The table displays the results both before and after FAC was applied. Static MFCCs were used as acoustic features. The configuration file used to obtain the feature vectors can be found in appendix A.2. Full covariance matrices were used in both the BIC and FAC algorithm. Note that the results from the modified version of BIC will vary slightly for each experiment, as the function `gaussmix.m` did not always return the same log likelihood score for the same data.

Table 5.7: Evaluation of speaker segmentation using modified BIC on RUNDKAST. Mismatch is the average mismatch between the change points accepted as correct and the true change points. Both the results before and after FAC are shown.

Criterion	PRC	RCL	F-score	Mismatch
Mod BIC:	68.2 %	82.7 %	74.4 %	341 ms
Mod BIC FAC:	75.3 %	79.6 %	77.1 %	339 ms

The results from using the modified BIC algorithm with the HUB-4 1997 data are displayed in table 5.8. The results are compared to the results obtained by [1] with the same data. When using the evaluation criterions mentioned in section 4.2.3 it was considered to be 594 change points in the database. [1] considered 515.

Table 5.8: Evaluation of speaker segmentation on HUB-4 1997. Results from proposed method are compared with results obtained by [1] with the same data.

Criterion	PRC	RCL	F-score
Mod BIC	65.4 %	78.4 %	71.3 %
Mod BIC FAC	70.3 %	75.5 %	72.8 %
Ajmera	68 %	65 %	67 %

The results from the original BIC algorithm are presented in table 5.9. Different values of λ, in equation (2.17), are displayed to show the effect on the precision and recall. The results were obtained with a minimum window of 2 seconds, maximum window of 8 seconds, window increment of 1 second and full covariance matrices. Static MFCCs were used, and FAC was not applied.

Table 5.9: Evaluation of speaker segmentation with original BIC on RUNDKAST.

Criterion	PRC	RCL	F-score	Mismatch
BIC $\lambda = 1.45$	68.7 %	82.9 %	74.5 %	401 ms
BIC $\lambda = 1.5$	73.3 %	77.8 %	74.8 %	367 ms
BIC $\lambda = 1.55$	76.9 %	74.0 %	74.8 %	342 ms
BIC $\lambda = 1.6$	79.9 %	72.3 %	75.1 %	318 ms
BIC $\lambda = 1.65$	81.6 %	69.1 %	74.1 %	314 ms

The best results obtained when using diagonal covariance matrices are displayed in table 5.10. In the modified version of BIC, the results were obtained using a minimum window of 1 second, a maximum window of 7 seconds and an increment of 0.5 seconds. In the original BIC, the results were obtained with a minimum window of 2 seconds, maximum window of 8 seconds and an increment of 1 second. λ was set to be 5.5. Static MFCCs were used, and FAC was not applied.

Table 5.10: Evaluation of speaker segmentation when using diagonal covariance matrices.

Criterion	PRC	RCL	F-score	Mismatch
Mod BIC, diag Σ	71.1 %	60.8 %	65.2 %	275 ms
BIC $\lambda = 5.5$, diag Σ	76.3 %	69.7 %	71.9 %	381 ms

The results when including pitch in the feature set are shown in table 5.11. The table displays the results when using both normalized and not normalized LL scores for the pitch features, for each of the two pitch extraction algorithms used. For the pitch algorithm from Edinburgh speech tools, the minimum and maximum fundamental frequency were set to respectively 50 Hz and 600 Hz. The α in equation (4.3) was set to 0.8. The modified version of BIC method was applied with a minimum window of 2 seconds, maximum window of 8 seconds, window increment of 1 second and full covariance matrices.

Adding delta coefficient turned out to worsen the results and changing the threshold for acceptance of change points in the modified version of BIC did not improve the results either. Both an automatic updated threshold and a static threshold were tested.

Table 5.11: Evaluation of speaker segmentation with inclusion of pitch in the feature set. The results when using normalized and not normalized LL scores for the pitch features are shown, for each of the two pitch extraction algorithms used.

Criterion	PRC	RCL	F-score	Mismatch
Praat FAC	75.3 %	79.3 %	76.8 %	326 ms
Praat norm FAC	77.1 %	78.0 %	76.9 %	330 ms
Edinburgh FAC	76.2 %	77.7%	76.6 %	338 ms
Edinburgh norm FAC	79.0 %	75.6 %	76.7 %	324 ms

Chapter 6

Discussion

This chapter includes a discussion of the results presented in chapter 5. Section 6.1 treats the results from the pause detection and the accompanying problems. Section 6.2 discusses the results from speaker segmentation. This section also includes a presentation of the additional cost it takes to use full covariance matrices instead of diagonal covariance matrices. Finally, section 6.3 gives a suggestion to further work.

6.1 Pause detection

6.1.1 Experiments

As can be seen from table 5.1, the number of errors gotten from the different parameterizations was quite similar for the best mixture combinations. Still, the inclusion of delta coefficients in the MFCC vectors gave an improvement over using static MFCCs. The sound model was believed to need a higher number of mixture components than the pause model, as there was much more training data available for sound. However, this was not the case for the static MFCCs. In an application where it is important to minimize the risk of loosing sound when removing pauses, settings that allow more false rejections than false alarms should be chosen. If a low computational cost is desired, one should choose the lowest number of mixtures and the highest values of window length and window shift that gives adequate results. Based on this, the parameters evaluated in table 5.2 were chosen. The obtained results were promising, which implied that the acoustic models should be applicable.

The results from the recognition of pauses, presented in table 5.3, did not seem good at first sight. The insertion penalty was set such that there

was about as many false alarms as false rejections. However, when listening to the false alarms, it was clear that about 90 % of these segments actually were periods of silence, noise or silence followed by a breathing sound. By listening to the false rejections, it was estimated that half of them should be fairly easy to detect. The other half were believed to be harder to detect as these contained loud or unstable noise. Yet, similar segments were recognized correctly.

The insertion penalty was changed to -40 in order to reduce the number of false rejections. The number of false alarms was consequently increased, but it was expected that many of these segments were pauses. Table 5.4 shows the scores before and after re-labeling of the false alarms. It should be noted that some transcribers used the label for breathing sound together with the label for short pause for the same type of segments that other labeled as a long pause. Such segments were accepted as correct detected pauses in the modified transcription. The high precision implied that the models were capable of detecting many pauses and that the biggest problem laid in the transcription. Yet, the number of missed pauses was unknown. Most of the false alarms that were left after the re-labeling contained fading music followed by a period of silence. Some also contained only music and some were filled pauses with content like ringing phones or background speech. The latter has been defined as sound in this thesis.

Proposed pauses of duration longer than 0.9 seconds were kept in the results. A long pause should, according to the transcription rules, be longer than 1 second since shorter pauses more often occurs as natural breaks in speech. In spite of that, some of the transcribed pauses in RUNDKAST were shorter. The minimum length of 0.9 seconds was therefore set. This also allows some mismatch in the recognized segment boundaries of pauses close to 1 second. With the insertion penalty of -40 there were 95 pauses in the results with duration between 0.9 and 1 second. Nine of these were labeled pauses, the rest were false alarms. By only allowing pauses longer than 1 second, one could have reduced the number of proposed false alarms considerably.

The new labeling of pauses in the HUB-4 1997 transcription gave promising results. However, since the labeling was performed by looking at the audio signal, pauses may have been missed. Especially pauses containing noises were hard to find. No reported results of pause detection on this corpus, which can be used for comparison, have been found. This may be due to the fact that labeling of pauses missed in the transcription.

Figure 5.1 presented the investigation of whether the models were applicable. With satisfying models it should be possible to separate the LL scores obtained with the sound model from the LL scores obtained with the

pause model. The plot showed, on the other hand, that the sound models were modeling pauses almost as well as the pause models. The reason may be that the training data for sound contained a good deal of longer pauses. The new training set therefore tried to reduce the risk of having long pauses in the training data for sound. As an example it was discovered that there was often a pause after music segments. The ends of music segments were therefore excluded. A few more pauses were included in the training set. More representative data was believed to give an improvement of the pause model. From table 5.6, it was clear that the numbers of false alarms and false rejections were about the same as with the first training set. When listening to the false alarms obtained with the new models, it turned out that the results had become poorer as several long segments of music were recognized as pauses. The reason for this is not known.

Training of models with a lot of data is time consuming. To reduce the amount of elapsed time, quite long window lengths and frame shifts were used for parameterization. Important information may have been missed because of this, and there might have been non-stationary content in the frames. In case of the latter, the assumption when calculating MFCCs is not valid.

6.1.2 Problems with the transcriptions

Because of the large amount of long pauses that not were labeled in the transcription for RUNDKAST, the evaluation of the recognition was problematic. In addition to getting unreliable results, not knowing the real number of false rejections made it hard to set the value of the insertion penalty in HVite. This parameter is useful to control the number of false alarms versus false rejections. As mentioned earlier, it is sometimes preferred with more false rejections than false alarms. The transcription may also have contributed to errors in the models if a lot of silence was included in the training set for sound. The reason for the poor transcription of pauses may be that the transcribers concentrated on labeling of speech, and therefore were inconsistent in their labeling of other events. As several people contributed to the transcription, things may have been handled differently. Some aspects with the transcription that caused problems in the pause detection are listed below.

- According to the transcription rules, a long pause should have a duration longer than 1 second. However, segments down to 0.615 seconds were marked as a long pause. The segment length therefore had to be registered, and if a labeled pause was shorter than 0.9 seconds it was not treated.

- Long pauses should have been separated out as segments in the transcription, but this was not always done. Segments containing both sound and a long pause were not included in the training set. This made fewer pauses in the set than it could have been. Enough representative training data is necessary to obtain accurate models.

- Many instances of the false rejections had similar content as found among the false alarms and the correct accepted pauses. This may be due to inconsistency in the labeling. For example, segments with a short pause followed by an inhalation sound did often have a duration longer than one second. Sometimes these segments were labeled as a long pause and other times they were labeled as exactly a short pause and inhalation.

Also in the transcription of the HUB-4 1997 data, the information about pauses was insufficient. Some pauses were marked as *inter segment gap*, but many unmarked pauses longer than 1 second were discovered. Some of the pauses that had been labeled were marked with incorrect segment boundaries. No rules for labeling pauses were found in the description of transcribing routines, so this might not have been a prioritized task in the transcription of the database.

6.2 Speaker segmentation

6.2.1 Experiments

The length of the speaker segments in the evaluated broadcast news varied from less than a second to several minutes. Short segments are often found in debates, while longer segments are common in for example read news. Because of the short segments, the minimum window size in the metric-based algorithms must be quite small. This makes it hard to get reliable models of the data. Despite that, satisfactory results were obtained as shown in table 5.7. The false alarm compensation increased the F-score with a few percents, and made the precision and recall more even. The average mismatch between segment boundaries accepted as correct and the boundaries in the reference transcription was 339 ms. As there often is a period of silence between two speaker segments, a change point can correctly be assigned in both the beginning and the end of this silence. Consequently, the mismatch obtained from the evaluation might sometimes have been a little larger than what could have been defined as correct. Experiments on adding delta coefficients to the MFCCs resulted in a worsening of the results. This may not be a

surprise as speech from the same speaker also contains temporal changes. It was also found out that the threshold of zero was the best choice in the modified version of BIC.

The modified version of BIC was first presented by Ajmera et al. [1]. In this thesis different parameters were chosen. Among other factors, full covariance matrices were used in this theses, while [1] used diagonal matrices. From table 5.8 it was clear that the method implemented in this thesis gave about 5 % increase in the F-score compared to [1]. In the evaluation of the results, all speaker changes with a mismatch less than 1.5 seconds were accepted. If this boundary is reduced, the F-score will be poorer, and the mismatch will be better. Ajmera et al. did not report any value of the mismatch, and it is therefore not known whether they were stricter with the acceptance of change points or not.

Table 5.9 shows that the original BIC method was better than the modified BIC when FAC not was used. With the original method, it was easier to control the number of false alarms versus false rejections. Therefore the FAC algorithm did not have to be applied. From table 5.10, one can see that in the modified version of BIC, full covariance matrix gave a significant improvement over diagonal covariance matrices. There was also an improvement in the original BIC, although this was smaller. The improved results indicate that there was some correlation between the MFCC vectors.

By comparing the results in table 5.11 with the results in table 5.7, it was clear that the inclusion of pitch in the feature set gave almost unchanged results. The reason may be that the estimates were not good enough or that the pitch varied too much for the same speakers. All four experiments gave about the same results. An investigation of the change points showed that the inclusion of pitch occasionally changed the BIC scores from below zero to above zero, and vice versa. It is believed that when this happened, often another BIC score in the same window was chosen as a change point. This would not affect the resulting F-score, but only contribute to a small change in the mismatch. In the experiment with normalized LL scores for the pitch values extracted with Praat, the F-score was lower than both the precision and recall. This was because of the evaluation method used where the average of the scores from each file was taken. The performance of the pitch detection depends on the maximum and minimum value of the fundamental frequency. This could be more accurate if one knew the gender of the person talking. An idea could be to do a gender classification before the speaker segmentation.

It turned out that a maximum window size of 8 seconds gave the best results. If the window length was increased, the results became poorer and the processing time increased. The window length was well suited for these experiments, but it was short compared to e.g. [25], which used 20 seconds.

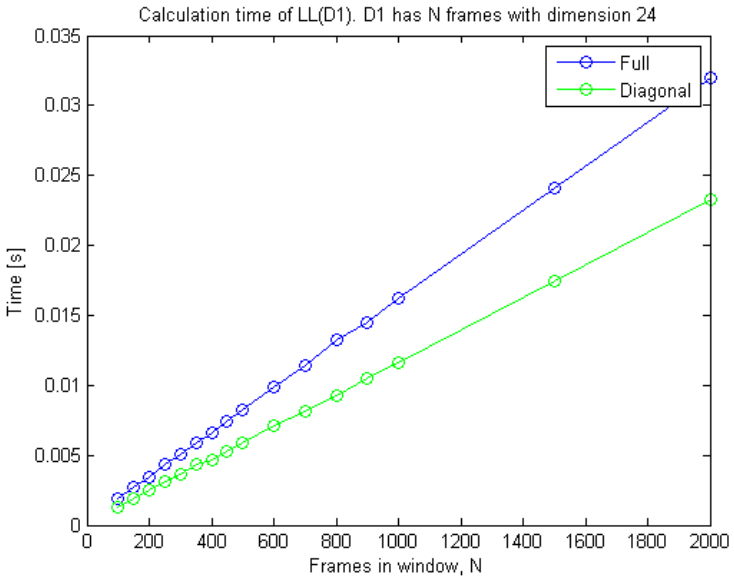
Duets often contributed to false rejections, as these speaker segments were very short, typically less than a second. In the test set from RUNDKAST, 9.7 % of all segments had a duration shorter than 2 seconds. When using a window length of 2 seconds, these were hard to detect. False rejections did also occur in cases with a sequence of the speakers $[a, b, a]$, where speaker b only contributed with a short interruption. The change to speaker b will most likely be missed because of the short duration of the speaker segment, and therefore speaker a, talking for the second time, will probably be missed as well. False alarms are likely to occur when the window is too short to give a reliable model of the data. Many false alarms were also introduced by the non-speech elements, but these were not included in the evaluation. Non-speech sounds will occur frequently in broadcast news, but they were chosen not to be considered as it will be easier to compare with others' results when only speech is treated. However, changes from speech to non-speech and vice versa were included. Most of these change points were successfully detected.

The test set consisted of parts from several different files. From file segment to file segment the F-score varied from 68.4 % to 88.6 %, after the FAC routine. The performance was therefore dependent on the speakers and/or background conditions. In the file with the lowest score, 25.7 % of the segments were shorter than 2 seconds and therefore hard to detect.

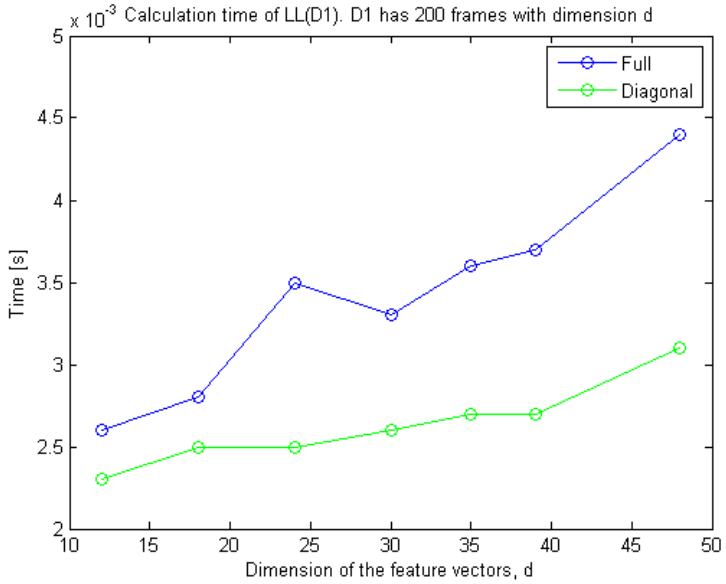
6.2.2 Full versus diagonal covariance matrices

When using diagonal covariance matrices for modeling the Gaussian distribution, it is assumed that the elements in the acoustic feature vectors are independent. This is approximately true when using MFCCs. Full covariance matrices are necessary to model correlated feature vectors, but the complexity of the algorithm increases when using full instead of diagonal covariance matrices. For d-dimensional feature vectors, a full covariance matrix requires d^2 parameters, while a diagonal matrix only requires d parameters. The difference in running time between using full and diagonal covariance matrix for modeling data with a single mixture GMM is displayed in figure 6.1. The plot shows the running time of the calculation of a LL score. Both the number of frames in the data window and the dimension of the feature vectors have been varied.

As seen from figure 6.1, the use of full covariance matrices was more costly than the use of diagonal matrices, and the time gap increased with the dimension of the data. In the modified version of BIC, LL scores of two data windows modeled by single mixture model were calculated for each time step. In addition a GMM with two mixtures was calculated for time each step. In the original BIC, LL scores of three data windows modeled by



(a) Varying the number of frames in a window. The dimension of the feature vectors is 24.



(b) Varying the dimension of feature vectors. The number of frames in the window is 200.

Figure 6.1: Running time for calculation of the LL score of data modeled by a single mixture model, varying window size and feature vector dimension. Green and blue graph shows the running times using diagonal and full covariance matrices respectively.

a single mixture model were calculated per time step. The number of BIC measurements in a file depends on the number of change points. A rough estimate is that in 15 minutes of audio, there are 150 000 points in time, t, where the BIC distance is calculated. The running time of the function used for the two mixture model, `gaussmix.m`, is shown in figure 6.2.

The scaling in figures 6.1 and 6.2, shows that the calculation of the LL score for the two mixture GMM dominated the running time. Thus, the cost of using full covariance matrices was larger for the modified version of BIC.

The difference between using full and diagonal covariance matrices when calculating a KL2 distance is displayed in figure 6.3. Both the number of frames in the window, and the dimension of the feature vectors have been varied. As seen in the figure, the difference in running time was constant with respect to the number of frames in the window, but it was increased with an increasing dimension of feature vectors. In the false alarm compensation, a maximum of 400 KL2-distances were calculated for each proposed change point, so this algorithm will contribute less to the total running time.

6.2.3 Comments to the transcriptions

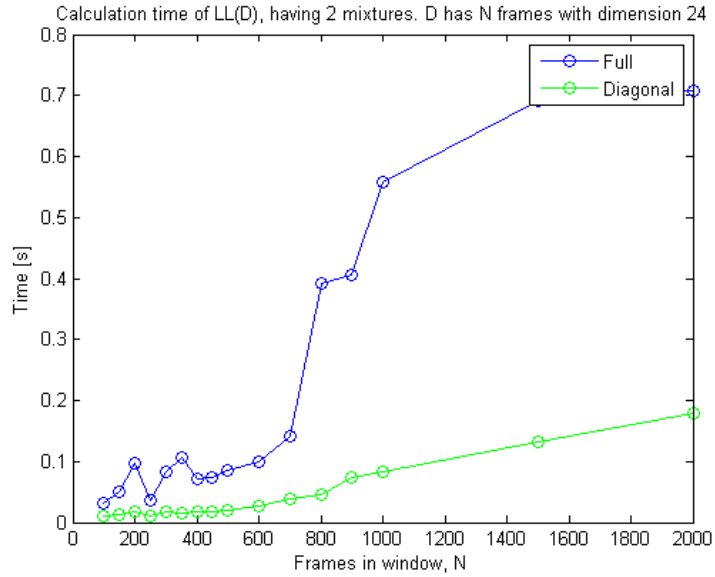
The transcriptions of the two corpora were believed to be good when it came to labeling of speaker changes. Even so, a few comments can be made. In the transcriptions belonging to RUNDKAST there was by coincidence found unlabeled occurrences of people talking simultaneously. Some errors may have occurred because of this, but since the examples found had quite short duration it is not believed to be a significant problem.

In both RUNDKAST and the HUB-4 1997 data, segments containing foreign speech were not transcribed. As these segments sometimes contained several people talking, music segments or similar, they had to be listened to in order to get a correct evaluation. This led to some more work with the evaluation, but the results were not affected.

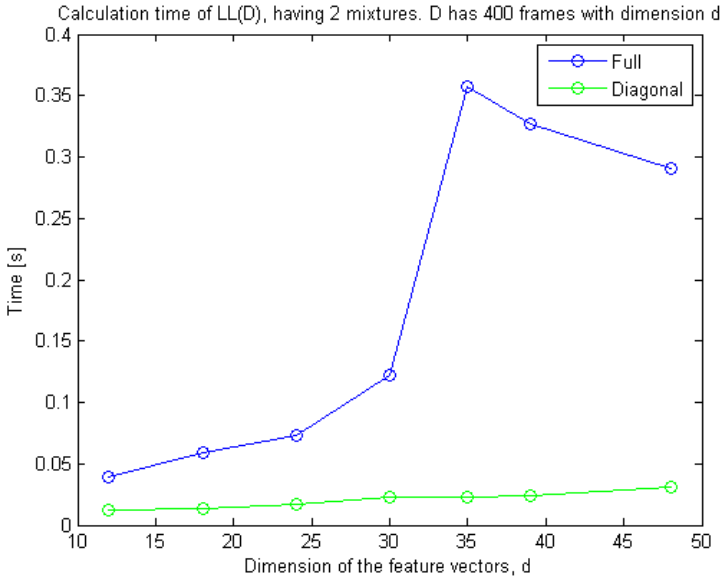
6.3 Further work

More work remains in order to have a tool that can extract a complete event log of broadcast news. First of all, the transcription of the database used, RUNDKAST, must be quality checked. Especially pauses are poorly documented. To able to find a properly working method for pause detection, the reference transcription must be correct.

False alarms that arose from segments with music or other non-human sounds in the speaker segmentation were not considered in this task. A

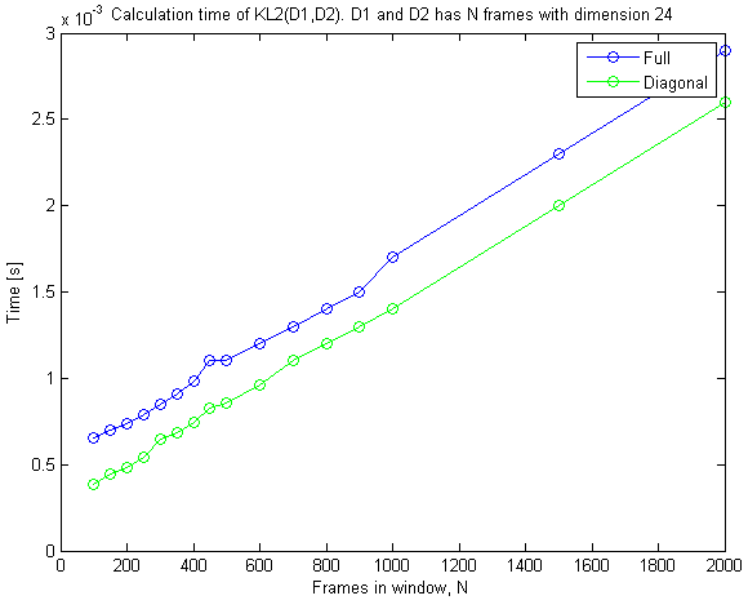


(a) Varying the number of frames in a window. The dimension of the feature vectors is 24.

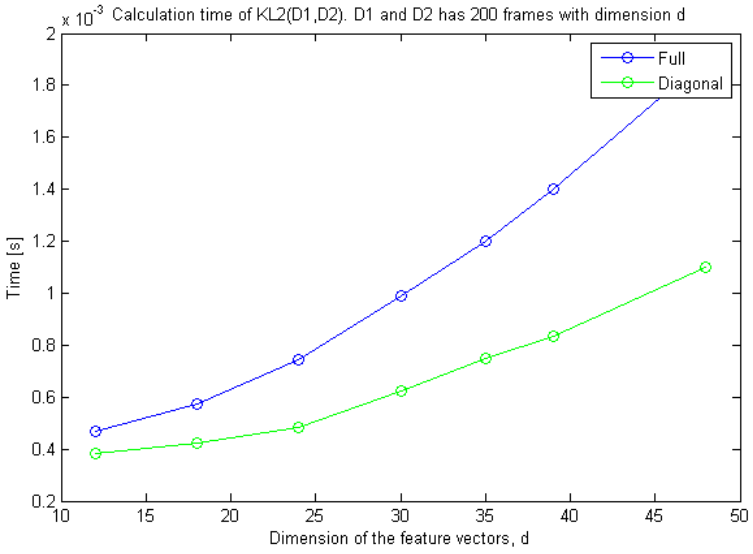


(b) Varying the dimension of feature vectors. The number of frames in the window is 400.

Figure 6.2: Running time for calculation of the LL score of data modeled by a two mixture model, varying window size and feature vector dimension. Green and blue graph shows the running times using a diagonal and full covariance matrices respectively.



(a) Varying the number of frames in a window. The dimension of the feature vectors is 24.



(b) Varying the dimension of feature vectors. The number of frames in the windows are 200.

Figure 6.3: Running time for calculation of one KL2 distance, varying the window size and feature vector dimension. Green and blue graph shows the running times using a diagonal and full covariance matrices respectively.

solution to the problem is to perform speech/non-speech segmentation before the speaker segmentation. It is in this case important not to loose any speech segments, something that may be difficult when dealing with segments containing overlapping music and speech. In other tasks one may want to find all parts in a file that contain music, and then such overlapping segments should be classified as music.

A follow-up to the speaker segmentation is to perform speaker clustering, where segments spoken by the same speaker are grouped together. This can later be used for adaption of acoustic models, which can improve the speech recognition rate. It can also be used for identification of speaker roles, which can make it easier to determine different topics and parts in a story [26]. The clustering can be done by first detecting changes in an audio stream, and secondly cluster the similar speaker segments together.

Other information that may be useful in an event log is the type of recording. Bandwidth measurements can be used to find out whether one has a studio recording, telephone recording or radio link.

Chapter 7

Conclusion

This thesis has started the process of making an event log of broadcast news. Experiments on detection of long pauses and speaker changes have been performed. The corpus RUNDKAST has been used for experimenting, and the HUB-4 1997 data has been used for comparative tests.

In the pause detection, the recognizer first got defined segments with the task to decide whether they contained a long pause or sound. Experiments were performed with different audio parameterizations and numbers of mixture components in the GMMs. It became clear that including delta coefficients in the feature vectors gave an improvement over using static MFCCs. A number of different mixture combinations gave about the same results, but 32 mixtures for the sound model and 16 mixtures for the pause model were used in further experiments. The classification showed promising results, with an F-score of 98.8 %.

When performing recognition instead of classification, it became clear that several pauses in both RUNDKAST and HUB-4 1997 were not labeled in the transcriptions. With an insertion penalty of -40, a precision of 16.6 % and a recall of 72.6 % were obtained with the test set from RUNDKAST. However, the false alarms were listened to and re-classified as correct detected pauses if they contained noise or silence. Consequently, the precision increased to 92.1 %. It was not known how many long pauses that were really missed, resulting in an unknown recall. An attempt on labeling all pauses in the HUB-4 1997 data was done. With the modified transcription, an F-score of 81.7 % was obtained. This is a promising result, but it is possible that unlabeled pauses still exist in the transcription, as the labeling was performed by only looking at the audio signal.

In the speaker segmentation, experiments were done with different algorithms and parameters. The modified version of BIC resulted in an F-score of 77.1 % after the false alarm compensation was run. The average mis-

match between accepted change points and the boundaries in the reference transcription was 339 milliseconds. The algorithm was also tested with the HUB-4 1997 data and an F-score of 72.8 % was obtained. For comparison, [1] reported an F-score of 67 % with the same data. The original BIC algorithm gave the best results without use of FAC, with an F-score of 75.1 %. From experiments it became clear that delta coefficients in the MFCC vectors gave a worsening of the results. Full covariance matrices gave an improvement over diagonal covariance matrices and the improvement was largest in the modified version of BIC. However, full covariance matrices also lead to increased computational complexity. Including pitch did not contribute to any improvement of the results.

In further work, a review of the transcription of RUNDKAST must be performed in order to be able to get trustworthy results from the pause detection. How to treat the non-speech elements so they do not affect the speaker segmentation should also be investigated. To complete the event log one should collect all speech from the same speakers by doing speaker clustering. Later one can also do speaker identification as many of the speakers in broadcast news occur frequently such as announcers and well known politicians. Detection of the recording type should also be done by bandwidth measures.

References

[1] J. Ajmera, I. A. McCowan, and H. Bourlard. BIC Revisited for Speaker Change Detection. Technical report, IDIAP-RR 39, Martigny, Switzerland, 2002.

[2] MIT. MIT Lecture Browser. URL `http://web.sls.csail.mit.edu/lectures/`. Date Accessed: 10.05.2011.

[3] C. Chelba, T.J. Hazen, and M. Saraclar. Retrieval and browsing of spoken content. *IEEE Signal Processing Magazine*, 25(3):39–49, May 2008.

[4] Jonathan Foote. An overview of audio information retrieval. *Multimedia Systems*, 7(1):2–10, January 1999.

[5] Chuck Wooters, James Fung, Barbara Peskin, and Xavier Anguera. Towards Robust Speaker Segmentation: The ICSI-SRI Fall 2004 Diarization System. In *Fall 2004 Rich Transcription Workshop (RT-04)*, Palisades, NY, 2004.

[6] T. Hain, S. E. Johnson, A. Tuerk, P. C. Woodland, and S. J. Young. Segment Generation and Clustering in the HTK Broadcast News Transcription System. In *1998 DARPA Broadcast News Transcription and Understanding Workshop*, pages 133–137, Lansdowne, VA, 1998.

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

[8] Kasper Jørgensen, Lasse Mølgaard, and Lars Kai Hansen. Unsupervised Speaker Change Detection for Broadcast News Segmentation. In *in Proc. EUSIPCO*, 2006.

[9] D. A. Reynolds and P. Torres-Carrasquillo. The MIT Lincoln Laboratory RT-04F Diarization Systems : Applications to Broadcast Audio and

Telephone Conversations. In *Fall 2004 Rich Transcription Workshop (RT-04)*, Palisades, NY, November 2004.

[10] Matthew A. Siegler, Uday Jain, Bhiksha Raj, and Richard M. Stern. Automatic Segmentation , Classification and Clustering of Broadcast News Audio. In *Proc. DARPA Speech Recognition Workshop*, pages 97–99, Chantilly, VA, 1997.

[11] Lie Lu, Hong-Jiang Zhang, and Stan Z. Li. Content-based audio classification and segmentation by using support vector machines. *Multimedia Systems*, 8(6):482–492, April 2003.

[12] Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, Valtcho Valtchev, and Phil Woodland. The HTK Book, December 2006. URL `http://htk.eng.cam.ac.uk/prot-docs/htkbook.pdf`.

[13] Zbynìk Tychtl and Josef Psutka. Speech Production Based on the Mel-Frequency Cepstral Coefficients. In *Proceedings of Eurospeech*, pages 2335–2338, Budapest, 1999.

[14] Xuedong Huang, Alex Acero, and Hsiao-Wuen Hon. *Spoken Language Processing*. Prentice-Hall, Inc., 2001. ISBN 0-13-022616-5.

[15] Julian James Odell. *The Use of Context in Large Vocabulary Speech Recognition*. PhD thesis, University of Cambridge, 1995.

[16] Yannis Stylianou, Olivier Capp, and Eric Moulines. Continuous Probabilistic Transform for Voice Conversion. *IEEE Transactions on Speech and Audio Processing*, 6(2):131–142, 1998.

[17] NTNU. RUNDKAST - A Norwegian Broadcast News Speech Corpus. URL `http://www.iet.ntnu.no/projects/rundkast/`. Date Accessed: 02.02.2011.

[18] K. Boudahmane, M. Manta, F. Antoine, S. Galliano, and C. Barras. Transcriber. URL `http://trans.sourceforge.net/`. Date Accessed: 02.02.2011.

[19] NIST Spoken Natural Language Processing Group. URL `http://www.nist.gov/itl/`. Date Accessed: 18.05.2011.

[20] Mike Brookes. VOICEBOX: Speech Processing Toolbox for MATLAB. URL `http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html`. Date Accessed: 07.03.2011.

[21] Bin Zhang, Meichun Hsu, and Umeshwar Dayal. K-Harmonic Means-A Data Clustering Algorithm. Technical report, Software Technology Laboratory, HP Laboratories Palo Alto, HPL-1999-124, 1999.

[22] Simon King, Alan W. Black, Paul Taylor, Richard Caley, and Rob Clark. Edinburgh Speech Tools Library System Documentation Edition 1.2, 2003. URL `http://www.cstr.ed.ac.uk/projects/speech_tools/manual-1.2.0/`. Date Accessed: 15.04.2011.

[23] Yoav Medan, Eyal Yair, and Dan Chazan. Super Resolution Pitch Determination of Speech Signals. *IEEE Transactions on Signal Processing*, 39(1), 1991.

[24] Paul Boersma. Accuraete Short-Term Analysis of the Fundamental Frequency and the Harmonics-to-Noise Ratio of a sampled sound. In *IFA Proceedings*, volume 17, pages 97–110. Institute of Phonetic Sciences, University of Amsterdam, 1993.

[25] J. Ajmera, I. McCowan, and H. Bourlard. Robust Speaker Change Detection. *IEEE Signal Processing Letters*, 11(8):649–651, August 2004.

[26] H. Meinedo and J. Neto. Audio segmentation, classification and clustering in a broadcast news task. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2:II–5–8, 2003.

[27] T. Kemp, M. Schmidt, M. Westphal, and A. Waibel. Strategies For Automatic Segmentation of Audio Data. In *Proc. ICASSP*, volume 3, pages 1423 –1426, July 2000.

[28] Ida Onshus. Computer Assisted Pronunciation Training. Technical report, Department of Electronics and Telecommunications, NTNU, December 2010.

[29] P. Jackson. readmfcc.m. URL `http://personal.ee.surrey.ac.uk/Personal/P.Jackson/eem.ssr/lab2/readmfcc.m`. Date Accessed: 03.03.2011.

Appendices

Appendix A

Parametrization

A.1 Configuration file for pause detection

The configuration file used for parameterization of the audio waveforms into MFCCs, for the pause detection is shown below.

```
# Acoustic parameterization

# Input waveform file format (16 kHz 16 bit wav format)
 SOURCEKIND     = WAVEFORM
 SOURCEFORMAT   = WAV
 SOURCERATE     = 625
 ZMEANSOURCE    = TRUE  # To avoid DC offset

# Parameterization (20 msec frame shift, 40 msec frame window)
 TARGETKIND     = MFCC_0_D_A_Z
 TARGETFORMAT   = HTK
 TARGETRATE     = 200000
 WINDOWSIZE     = 400000
 USEHAMMING     = TRUE
 PREEMCOEF      = 0.97
 USEPOWER       = FALSE
 NUMCHANS       = 26
 LPCORDER       = 12
 CEPLIFTER      = 22
 NUMCEPS        = 12

 SAVECOMPRESSED = FALSE
 SAVEWITHCRC    = FALSE
```

A.2 Configuration file for speaker change detection

The configuration file used for parameterization of the audio waveforms into MFCCs, for the speaker change detection is shown below.

```
# Acoustic parameterization

# Input waveform file format (16 kHz 16 bit wav format)
 SOURCEKIND     = WAVEFORM
 SOURCEFORMAT   = WAV
 SOURCERATE     = 625
 ZMEANSOURCE    = TRUE  # To avoid DC offset

# Parameterization (10 msec frame shift, 30 msec frame window)
 TARGETKIND     = MFCC
 TARGETFORMAT   = HTK
 TARGETRATE     = 100000
 WINDOWSIZE     = 300000
 USEHAMMING     = TRUE
 PREEMCOEF      = 0.97
 USEPOWER       = FALSE
 NUMCHANS       = 26
 LPCORDER       = 12
 CEPLIFTER      = 22
 NUMCEPS        = 24

 SAVECOMPRESSED = FALSE
 SAVEWITHCRC    = FALSE
```

Appendix B

Experiments on Speaker Segmentation

Below is the experiments performed when doing speaker segmentation listed.

In the modified version of BIC, the following experiments were done:

- Testing the effect of full versus diagonal covariance matrices.
- Testing the effect of different thresholds when using full covariance matrices; the original threshold of zero, a static threshold different from zero, and an automatic adjusted threshold that depended on the mean of the BIC scores around a possible change point. This was implemented as

```
meankl = sum(all_delta(times(i)-199:times(i)+200))/...
    length(all_delta(times(i)-199:times(i)+200));
if meankl>0
    thresh = alpha1*meankl;
else
    thresh = alpha2*meankl;
end
```

- Testing the effect of 1st order delta coefficients when using full covariance matrices and different types of thresholds.
- Testing the effect of 1st and 2nd order delta coefficients when using full covariance matrices and a threshold of zero.
- Testing the effect of different values of the parameters `min_window`, `inc` and `max_window`.

- Testing the effect of choosing the best proposed change point per window, versus choosing the first.
- Testing the effect of pitch as an acoustic feature in addition to the MFCCs.

In the original BIC algorithm, the following experiments were done:

- Testing the effect of full versus diagonal covariance matrices.
- Testing the effect of 1st order delta coefficients when using full covariance matrices.
- Testing the effect of different values of the parameters `min_window`, `inc`, `max_window` and `lambda`.

In the experiments for false alarm compensation, the following experiments were done:

- Testing the effect of different values of BIC versus KL2 as the distance metric.
- Testing the effect of full versus diagonal covariance matrices.
- Testing the effect of different values of the parameters `Tm`, `alpha_fac` and `minimum_window`.

Appendix C

Classification Results

Table C.1 presents the results from the classification between sound and long pauses. All the mixture combinations that gave the best results are included.

Table C.1: Evaluation of the sound/pause classification for different numbers of mixtures components, dynamic parameters, window lengths and frame shifts. All mixture combination that gave the best results are included

Experiment	FA	FR
D and A, WL = 40ms, FS = 20ms		
64/32 mix	1	1
32/32 mix	2	0
32/16 mix	1	1
16/32 mix	2	0
16/16 mix	2	0
Only D, WL = 40ms, FS = 20ms		
64/32 mix	1	1
D and A, WL = 30ms, FS = 15 ms		
64/32 mix	2	0
64/16 mix	2	0
32/32 mix	2	0
16/32 mix	2	0
16/16 mix	2	0
Static MFCCs, WL = 40ms, FS = 20 4/32 mix	1	4

Appendix D

MATLAB Code

This appendix includes the MATLAB scripts used for speaker segmentation. The MATLAB script `Readmfcc` from [29], has been used for reading the MFCCs extracted with HTK into MATLAB. Section D.1 presents the code used for the modified version of BIC. The code used for the original BIC method and for calculations with diagonal covariance matrices, has been included and commented out. Section D.2 presents the code that was changed when pitch was included in the feature set. Section D.3 presents the script used for false alarm compensation.

D.1 BIC

```
function BIC(fileName,start,stop)


% modified BIC method
% written by Ida Onshus, 03.2011.
% It utilises the function 'gaussmix' from the toolbox Voicebox
% by Mike Brooks copyright(c) 1997 (GNU General Public License)
% fileName = file to be processed
% start  = sample number to start BIC calculation (optional)
% stop  = sample number to stop BIC calculation (optional)

% Read MFCCs
ifn = strcat('../mfcc/',fileName,'.mfc')
readmfcc;

if nargin == 1 % if only fileName given, start from sample 1
    start = 1;
elseif nargin == 3 % if all inputs given, include samples 1:stop
```

```
    Y = Y(:,1:stop);
end


% Set initial parameters
a=start; % window D start
b=start + 400 -1; % window D end
min_window = 200;  %minimum window size of D1, D2 in frames
inc = 100; % window increment in frames
max_window = 800; % max window size of D (8 sec)
changes = []; %save change points

% If original BIC method used
%deltaK = nC+nC*(nC+1)/2; %for full cov-matrices
%lambda = 1.4;

%array containing likelihoods of window D1
y1 = zeros(1,length(Y));
%array containing all BIC values used for automatic threshold
all_delta = zeros(1,length(Y));

while b < length(Y(1,:)) % while b < number of frames in Y

    D = Y(:,a:b)';  % Window D
    addpath voicebox
    % 2 mixture gaussian model remove v if diagonal cov matrix
    [m,v,w,g,f,pp,gg]=gaussmix(D,[],[],2,'hfv');
    rmpath voicebox
    y =sum(pp); % log likelihood

    % If original BIC method used
    %D = Y(:,a:b)';
    %mu = mean(D,1);
    %sigma = cov(D,1);
    %N = size(D,1);
    %invers = inv(sigma);
    %y = 0;
    %for i=1:N
    %    y = y +(D(i,:)-mu)*invers*(D(i,:)-mu)';
    %end
    %y = -0.5*(y + N* log(det(sigma))+ N*nC*log(2*pi));

    % initializing arrays
    deltaL = zeros(1,b-a-2*min_window + 2); %BIC values
    times = zeros(1,b-a-2*min_window + 2); %time points
    counter = 0;

% all t in window D that assures a minimum length of D1 and D2
    for t=(a+min_window-1):(b-min_window)
```

```
counter = counter + 1; %count bic measures
% left window. Calculate if not exists
if y1(t) == 0
    D1 = Y(:,a:t)';
    mu1 = mean(D1,1); % mean
    sigma1 = cov(D1,1);% full covariance matrix
    % Calculate log likelihoods
    tmp = 0;
    invers1 = inv(sigma1);
    for i=1:size(D1,1)
        tmp = tmp +(D1(i,:)-mu1)*invers1*(D1(i,:)-mu1)';
    end
    y1(t) = -0.5*(tmp + size(D1,1)*log(det(sigma1))+...
        size(D1,1)*nC*log(2*pi));% nC=dim of MFCCs

    % if diagonal covariance matrix:

    % sigma1 = diag(cov(D1,1));
    % tmp = 0;
    % invers1 = 1./sigma1';
    % for i=1:size(D1,1)
    %    tmp=tmp + (D1(i,:)-mu1).*invers1*(D1(i,:)-mu1)';
    % end
    % y1(t)=-0.5*(tmp + size(D1,1)*log(prod(sigma1))+...
    %    size(D1,1)*nC*log(2*pi));


    invers1 = 1./sigma1';
    for i=1:size(D1,1)
       tmp = tmp + (D1(i,:)-mu1).*invers1*(D1(i,:)-mu1)';
    end
    y1(t) = -0.5*(tmp + size(D1,1)*log(prod(sigma1))+ ...
        size(D1,1)*nC*log(2*pi));

end

% right window. Calculate for each shift
D2 = Y(:,(t+1):b)';
mu2 = mean(D2,1);
sigma2 = cov(D2,1);

% Calculate log likelihoods
y2 = 0;
invers = inv(sigma2);
for i=1:size(D2,1)
    y2 = y2 +(D2(i,:)-mu2)*invers*(D2(i,:)-mu2)';
end
y2 = -0.5*(y2 + size(D2,1)* log(det(sigma2))+ ...
    size(D2,1)*nC*log(2*pi));
```

```
        deltaL(counter) = y1(t) + y2 - y; % modified BIC measure
        all_delta(t) = y1(t) + y2 - y;
        times(counter) = t;

        % If original BIC method used
        % deltaL(counter)=y1(t)+y2-y-0.5*lambda*deltaK*log(N);

        % if choosing the first possible changepoint:
        % if deltaL(counter) > 0
        %     break
        % end
    end %for

    [v, i] = max(deltaL); %best candidate per window


    % automatic calculations of thresholds

%     alpha1 = 50;
%     alpha2 = 2/alpha1;
%     meankl = sum(all_delta(times(i)-199:times(i)+200))/...
%         length(all_delta(times(i)-199:times(i)+200));
%     if meankl>0
%         thresh = alpha1*meankl;
%     else
%         thresh = alpha2*meankl;
%     end


    thresh = 0; %threshold
    if v <= thresh % if no change points
        b = b + inc; % increment the window length
    else
        t = times(i) % print proposed change points
        changes = [changes t]; %save change points
        a = t+1; % move window boudaries
        b= a + min_window*2 -1; % move window boudaries
    end

    if (b-a) > max_window % if window size exeeds maximum
        a = b-max_window;
    end
end %while

%print change points to file
fid = fopen('changes.txt', 'a');
fprintf(fid,'mfcc/%s.mfc\n',fileName(1:20));
```

```
fprintf(fid,'%f\n',changes);
fclose(fid);
```

D.2 Changes in the BIC-script for inclusion of pitch

```
% Additional code for the modified version
% of BIC, when combining MFCCs with pitch.
%
% written by Ida Onshus, 04.2011.



% read pitch values from file
fid = fopen(strcat('festival_pitch/',fileName),'r');
%fid = fopen(strcat('pitch/',fileName,'.wav.txt'),'r');
C = textscan(fid,'%s');
fclose(fid);


% Modelling Data. Only example of window D1 is shown

% Widow D1 - Mfcc
D1 = Y(:,a:t)';
mu1 = mean(D1,1);
sigma1 = cov(D1,1);
% Calculate likelihoods
tmp = 0;
invers = inv(sigma1);
for j=1:size(D1,1)
    tmp = tmp +(D1(j,:)-mu1)*invers*(D1(j,:)-mu1)';
end
y1(t) = -0.5*(tmp + size(D1,1)* log(det(sigma1))+...
    size(D1,1)*nC*log(2*pi));
% Widow D1 -pitch
D1_p = str2num(char(C{1}(a+1:t+1)));
D1_p = D1_p(D1_p≠0); % don't include undefined pitch
if length(D1_p) < 5 % must be minimum 5 pitch values
    y1_p(t) = 0; % set the pitch likelihood to zero
else
    mu1_p = mean(D1_p,1);
    sigma1_p = cov(D1_p,1);
    tmp = sum((D1_p-mu1_p).^2);
    y1_p(t) = -0.5*(tmp/sigma1_p + size(D1_p,1)*...
```

```
        log(2*pi*sigma1_p));
    %combine likelihoods
    y1_p(t) = alpha*y1(t) + (1-alpha)*y1_p(t);
end


% Calculation of BIC measure
% if all windows have enough defined pitch
% values, include pitch. If not, use only mfcc
if y_p≠0 && y1_p(t) ≠0 && y2_p≠0
    deltaL(counter) = y1_p(t) + y2_p - y_p...
        - 0.5*lambda*deltaK*log(N);
else
    deltaL(counter) = y1(t) + y2 - y- 0.5*lambda*deltaK*log(N);
end
```

D.3 False alarm compensation

```
% False alarm compensation
% written by Ida Onshus 03.2011


%Initialize parameters
Tm = 200; % Maximum window length in frames (= 2sec) if not
          % change points are closer
alpha_fac = 0.5; % adjustment parameter for threshold

% Read bic-change points from file
%fid = fopen('../BIC_changes','r');
fid = fopen('../thresh','r');
C = textscan(fid,'%s');
fclose(fid);
ch = 1; %counter for all change points


while ch < length(C{1})

    % read MFCCs for the current file
    ifn = strcat('../',char(C{1}(ch)))
    readmfcc;

    k = ch+1; % counter for change points in current file
    values= [];
    % read all changepoint in one file and add to values
    % new file when line starts with 'mfcc'
    while strncmp(C{1}(k), 'mfcc', 4) == 0 && k<length(C{1})
```

```
    values = [values str2num(char(C{1}(k)))];
    k = k+1;
end
ch = k;

%array for KL2 distances for each proposed change points
KLdist = zeros(1,length(values));
changepoint = []; % array with the resulting change points

% for all changepoints in current file
for i=1:length(values)
    % Find if maximum window length must be <2 sec because
    % change points are closer, or if in start/end of file
    if i == 1
       Tmax = min([Tm values(i)-1 values(i+1)-values(i)-1]);
    elseif i == length(values)
       Tmax = min([Tm values(i)-values(i-1)-1 ...
           length(Y)-values(i)-1]);
    else
       Tmax = min([Tm values(i)-values(i-1)-1 ...
           values(i+1)-values(i)-1]);
    end

    % set windows with length Tmax, find mean and covariance
    D1 = Y(:,values(i)-Tmax:values(i))';
    D2 = Y(:,values(i)+1:values(i)+Tmax)';
    mu1 = mean(D1,1)';
    mu2 = mean(D2,1)';
    sigma1 =cov(D1,1);
    sigma2 =cov(D2,1);

    %KL2 distance between speaker seg 1 and speaker seg 2
    KLdist(i) = 1/2*trace((sigma1-sigma2)*...
        (inv(sigma2)-inv(sigma1)))+ 1/2*trace((inv(sigma1)...
        + inv(sigma2))*(mu1-mu2)*(mu1-mu2)');

%%%%%%%%%%%%%%%% Calculation of threshold %%%%%%%%%%%%%%%%
   % Threshold is calculated from the mean of KL2 distances
   % around the proposed change points


    lw=100; % minimum window length

    % find boundaries for where window starts and stop
    if values(i)-Tmax <= lw
        start = lw + 1;
    else
        start = values(i)-Tmax;
    end
```

```
        if values(i)+Tmax + lw > length(Y)
            stop = values(i)+Tmax - lw;
        else
            stop = values(i)+Tmax;
        end

        % calculate KLdistances for each time point t in window

        ls = 1; % window shift
        k_KLdist = zeros(1,floor((stop-start+1)/ls));
        teller = 1;

        for t=start:ls:stop % for each window shift

            window1 = Y(:,t-lw:t)'; % left window
            window2 = Y(:,(t + 1):(t + lw))'; % right window
            k_mu1 = mean(window1)';
            k_mu2 = mean(window2)';
            k_sigma1 = cov(window1,1);
            k_sigma2 = cov(window2,1);

            k_KLdist(teller) = 1/2*trace((k_sigma1-k_sigma2)*...
                (inv(k_sigma2)-inv(k_sigma1))) + 1/2*...
                trace((inv(k_sigma1) + inv(k_sigma2))*...
                (k_mu1-k_mu2)*(k_mu1-k_mu2)');

            teller = teller +1;
        end

        maxim = max(k_KLdist);
        minim = min(k_KLdist);
        % threshold is the mean value of KL2 distances in the
        % window. mimimum and maximum value is removed.
        threshold = alpha_fac/(length(k_KLdist)-2)*...
            (sum(k_KLdist)-maxim-minim);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        % If the KL2 distance in proposed changepoint is larger
        % than threshold, keep the changepoint
        if KLdist(i) > threshold
            changepoint = [changepoint; values(i)];
        end

    end

end
```