NTNU
Norwegian University of
Science and Technology

Discriminating Music,Speech and other Sounds and Language Identification

Tommy Strømhaug

Master of Science in Computer Science
Submission date: August 2008
Supervisor: Arne Halaas, IDI
Co-supervisor: Will Archer Arentz, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

Investigate the possibilities to discriminate between music,speech and other sounds. Also investigate the possibilities for automatic language identification.

Assignment given: 15. January 2008
Supervisor: Arne Halaas, IDI

Abstract

The tasks : discriminating music, speech and other sounds and language identification have a broad range of applications in todays multilingual multimedia community. Both tasks gave a lot of possibilities regarding methods and development tools which also brings some risk. The Language Identification(LID) problem ended up with two different approaches. One approach was discarded due to poor results in the pre-study while the other approach had some promising potential but did not deliver as hoped in the first place. On the other hand, the music, speech discrimination was solved with great accuracy using 3 simple time domain features and Support Vector Machines(SVM). Adding 'other sounds' to this discrimination problem did complicate the problem but the final solution delivered great results using the enormous BBC Sound Effects library as examples of non speech and music. Both tasks were tried being solved using Gaussian Mixture Models(GMM) because of it's known great ability to model arbitrary feature space segmentations. The tools used were Matlab together with a number of different toolboxes explained further in the text.

Acknowledgements

I want to thank Laila Wang for her patience and for being a great mother and girlfriend. Gratitudes goes to my supervisor Dr. Will Archer Arentz for great support with all sort of problems and to prof. Arne Halaas for great support and for setting me in contact with Will. I also want to thank prof. Ian Nabney for his excellent support on Netlab and his help on numerous problems regarding machine learning in general.

List of Figures

List of Tables

Contents

Chapter 1

Speech Music Other Discrimination

In this chapter an approach for discriminating between music, speech an other sounds will be tried. The chapter starts with a simple time domain speech music discriminator and from there 2 different approaches to do a discrimination between music, speech and all 'other sounds' are tested. Some terms like GMM, SDC, MFCC, PLP etc. are not deeply explained until chapter 3 where they are used the most. Setting chapter 3 as 1 would destroy the chronological development of the language identification system. The theory of some fundamental methods like Fourier transform, windowing, RMS(root mean square) etc. will not be explained.

1.1 Introduction

In the last decade the problem of discriminating speech from music has gained interest. The solution is still open, but accuracies over 90 percent have been reached. The results depends on many factors, like corpus used, features and test methods. It is difficult to compare two experiments if both were not using the same corpus or collection of data because of noise and channel distortions. The applications of the discrimination between music and noise are numerous.

[56] proposed a real-time speech/music discriminator, which was used to automatically monitor the audio content of FM audio channels. The application could be to change channels during commercials. He claims 98 percent accuracy using different statistics of the Zero Crossing Rating which is very cheap to compute. In automatic speech recognition in real world domain it can be useful as shown in [58]. It should be possible to disable the input to the speech recognizer during the non-speech portion of the audio stream. They use 13 features from both time and frequency domain to construct the discriminator. The dataset is the same as used

in this work and they claim an error rate at 5.8 percent when used on a frame by frame basis(20 milliseconds) and 1.4 percent over a segment of 2.4 seconds. They used a Gaussian Mixture model in combination with a nearest neighbor classifier.

Another useful application is coding of audio. If an audio stream could be separated into pieces of speech and music one could use a low bit coder for speech and a hight bit coder for music to increase the bandwidth efficiency[45].The classifier used was Gaussian Mixture Model(GMM) and features were Mel Frequency MFCC an accuracy of up to 97.14% for music and 93.87% for speech were attained.

The task of discriminating music, speech and other sounds has, as far as the author know, never been tried before. There have been some experiments based on a reduction of the term 'other sounds' to animal sounds, environmental sounds and noise. This task depends on finding features that discriminate speech and music strongly, and it could result in a big failure as long as 'other sounds' mean all other sounds in the universe. The term 'all other sounds' is comprehensive and there will always exist sounds in this group that are acoustically equal to speech and music. One alternative is to reduce it to speech, music and environmental or naturally sounds, but also here data collection could be a problem.[41] did a discrimination between speech, music, animal sounds and environmental sounds, but he had samples from all classes.In [68] a discrimination of speech and non-speech was accomplished based on phoneme recognition, but both training and testing were done with collected data.

1.2 Corpus

To make a corpus that includes speech and music was at one point attempting, but the Music-Speech Corpus was chosen[59]. It has also been used in two different studies[58, 71]. The corpus contains a training part and a test part for both speech samples and music samples. Each sample is 15 seconds long and there exist 240 of them. The music samples are both vocal and non-vocal and spawn a large dimension of genres.

1.3 A First Attempt

If one considers fig 1.1 and fig 1.2 it is clear that the music waveforms and the speech waveforms have some differences. First, the speech show a distinct pattern of high energy states followed by low energy states while music has a more even distribution of energy, except for rap music which is a problematic class of music to discriminate from speech[17][33]. The waveforms also shows that the amplitude variations of speech is higher than that for music. The latter is not really true about rap music. So, just by

looking at the waveforms of speech and music it is clear that speech has more low energy frames than music due to the (silence intervals) and higher amplitude variations. The silence intervals alone seems to be an important feature, because music does not have the same pattern of silence intervals as speech. This could be enough to build a simple speech/music discriminator in the time domain with these features :

- The percentage of low energy frames and high energy frames
- The percentage of silence frames
- The variance of energy

The feature extraction was done in Matlab and a classifier was build based on LIBSVM[11],a Support Vector Machine(SVM) library, and the implementation WLSVM[18] which is a wrapper for LIBSVM for Weka[72]. Weka can be used a standalone tool, but one also has the alternative to use classes from Weka in their own implementations. The latter was the chosen alternative.

The initial time interval for feature extraction is 2 seconds, this means that a a classification is made based on 2 seconds of data. If the RMS for a frame was less than 0.0070 then it was defined as silence. Each frame was 2 milliseconds and a low energy frame was defined as less than 50% of the average RMS for that interval. At this point one can suspect that there exist some correlation between low energy frames and silence frames, because a silence frame is also a low energy frame.

The best core for this problem was a linear, epsilon was set to 0.001 and the cost factor was varied between 0 and 120.

The resulting accuracy was **89.1626%** computed with a 10-fold cross correlation and the confusion matrix is in table 1.1. Then the same algorithm was tested for feature extraction intervals of 1,3 and 4 seconds and one could see a slighly increase in accuray using 3 seconds intervals. An accuracy of **90,348%** was achieved. Not a big deal and proably not significant due to a low number of training samples.

Then the Matlab function was tuned to find the best value of low energy frames. A low energy frames was defined as a frame with 50 percent lower energy than the average. This percentage value was adjusted from 60 to 1 in small steps and reached a maximum at 15 percent. The frame values were also tested, but 20 milliseconds seemed to be near optimal.

Dealing with a linear core the cost factor is the major parameter to optimize. All values from 1 to 200 with a step of 1 was run, while the epsilon was set to 0.001, and and accuracy of **96,0345%** was achieved with the cost factor set to 58. The confusion matrix is found in table 1.2. This is quite good considering the simple features used.

music	speech	
356	50	music
38	368	speech

Table 1.1: Confusion matrix

music	speech	
281	9	music
14	276	speech

Table 1.2: Confusion matrix

1.4 The Features

Using the same settings the worth of each feature was investigated and the result is in table 1.3. This table tells that there probably were some correlation between the features. The maximum accuracy was 96 percent and the percentage of low energy frames alone gives an accuracy of 94 percent. The two other features contribute very little to this result in combination with low energy frames.

1.5 A Step Further

As seen it is possible to achieve a good performance in discriminating between music and speech by just using some simple features in the time domain. The discrimination between music and speech will further be explored, but also the possibility to perform a discrimination of music/speech and other sounds will also be considered. These other sounds could be for instance an explosion, a car driving by, bird songs or any environmental sounds. This will be a much harder task than just discriminating music and speech and it obvious that it can't not solved using the simple energy features above. Features that define both music and speech strongly, have to be found. Another problem is how to perform the classification? Some of the options are explained below.

Variance	72.6601
Percentage of silence frames	83.867
Percentage of low energy frames	94.5813

Table 1.3: Accuracy of features in isolation

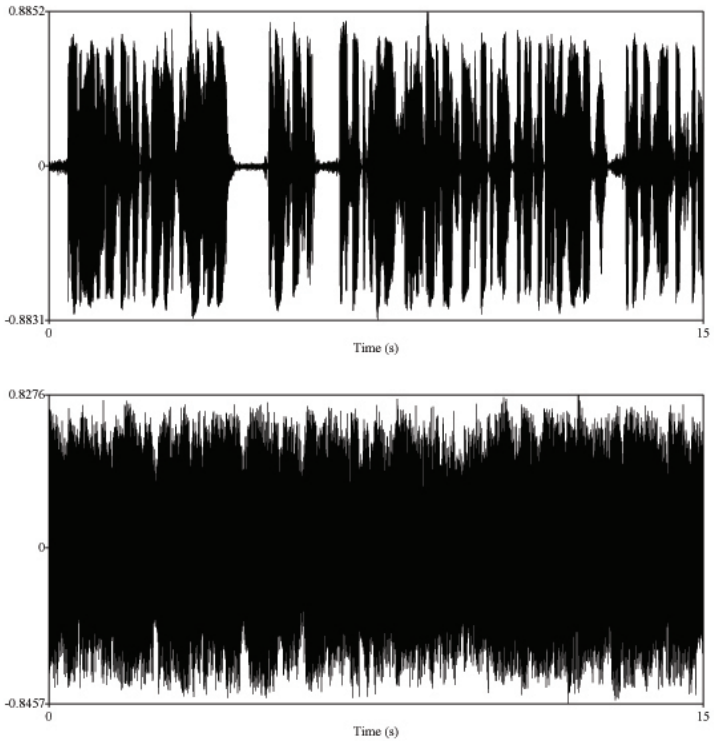

Figure 1.1: Top waveform is clean speech while bottom is a waveform of rock music

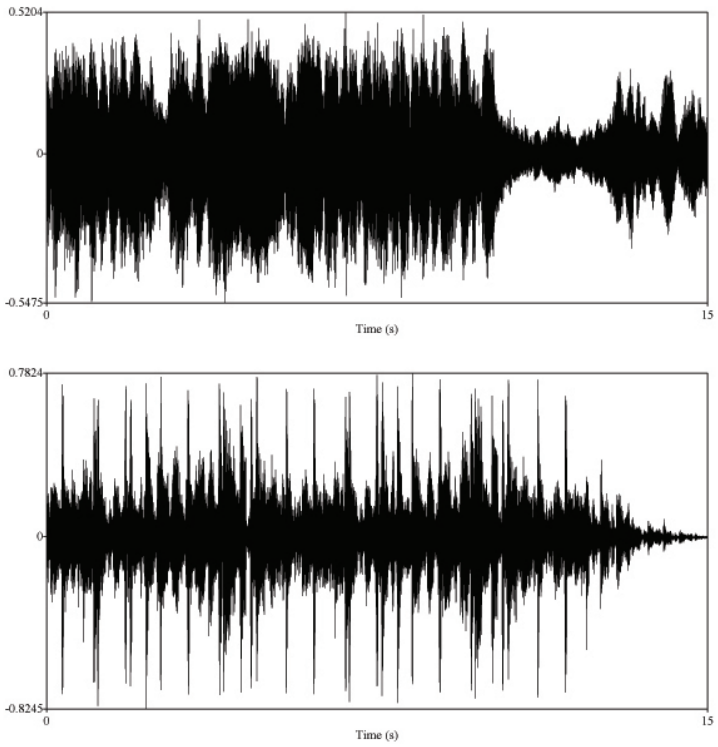

Figure 1.2: Top waveform is classical tune while bottom waveform is rap music

1.5.1 One Class Classifiers

A one class classifier's[57]goal is to construct a decision surface around the examples from the target class in order to distinguish between target objects and all the other possible objects, called outliers. A rejection rate is chosen during training so that a certain percentage of training patterns lies outside the constructed decision surface in order to take into account the possible presence of noise or outliers in the training set and to obtain a more precise description of the target class. There exist several one class algorithms, but only few will be considered here.

1.5.2 One Class SVM

A one-class classifier inspired by the two class SVM was proposed by [57]. The one-class classification problem is formulated to find a hyperplane that separates a desired fraction of the training patterns from the origin of the feature space F. If using the RBF kernel for the one class svm in libSVM[11] the are to parameters to deal with, namely the *gamma* and the *nu* parameters. *nu* is a lower bound of the fraction of Support vectors, and an upper bound of the fraction of outliers (the '-1' class). For *nu*=1 one gets a Parzen windows estimator, while for 0¡nu ¡ 1 one gets a thresholded decision function. For instance, when nu equals 0.2 up to 20 percent of the instances could be considered as outliers and at the same time it is the lower bound of the fractions of support vectors. The gamma parameter can simply be explained as a smoothness factor of the hyperplane. In fig 1.3 and fig 1.4 one can see two examples with 2D vectors with different nu and gamma. The real problem is how to adjust the parameters when one has high dimensional data and proper visualization is impossible. Using cross validation is a waste of time because the accuracy will reflect the *nu* parameter. A simple test with high dimensional data showed how difficult the parameter adjusting really was. Using one class SVM one would train a model for speech and a model for music. If the test instances do not fit the models of speech and music, it means it is something else.

1.5.3 Gaussian Mixture Models

The details of Gaussian Mixture Models are found in Language Identification chapter. The idea is to train GMMs with data from each class. In this case it will be one GMM trained with data for music, and a second one trained with data for speech. Then one has to use probabilities to decide if it is music, speech or other sounds. An attempting strategy would be to consider the probability densities: if the probability for music is low and the probability for speech is low, its probably something else.

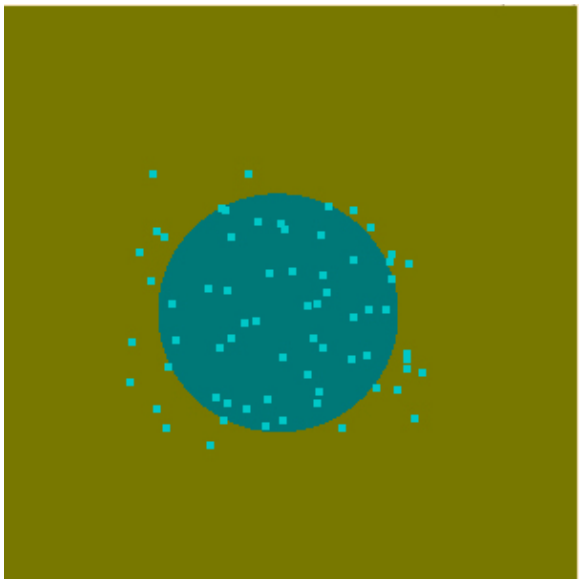

Figure 1.3: One class svm : nu=0.4 and gamma=0

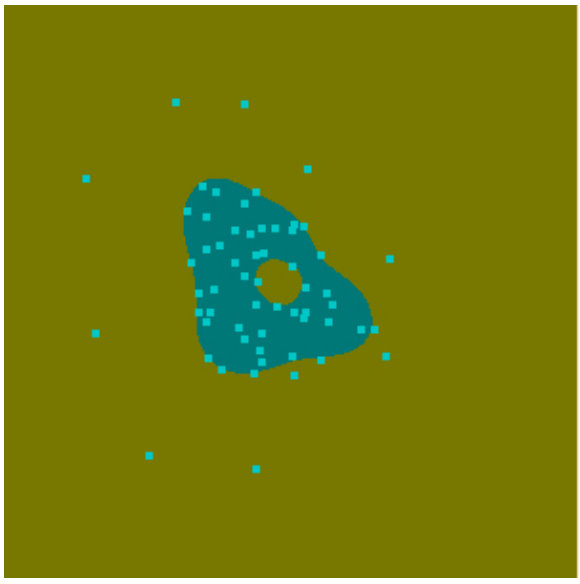

Figure 1.4: One class svm : nu=0.3 and gamma=60

1.5.4 2 Class SVM with calculated probabilities

Support Vector Machines (SVMs)[67] have become one of the most popular tools for discriminative classification of static data. SVM classification of dynamic (continuous) is not very successful using the traditional kernels. See section refsec:svm for details on this matter. The libSVM tool[11] has the ability to output probability predictions. One possibility could be to train a 2 class model with music and speech classes and then use the predictions to decide if it is music, speech or anything else. One could guess that a 50-50 probability with a +/- 2 percent could be the anything else.

1.6 Tools And Implementation

The choice of classifier became GMM after some research and the main reasons were because of it's known ability to model speech data and the shortcomings of SVM for the same task as explained in the beginning of section refsec:svm. The tools used for implementation is Matlab and Netlab Toolbox for Matlab[1] which contain implementations of several machine learning algorithms. Other machine learning toolboxes were also considered, but the final decision was based on the simplicity of Netlab. After some experiments with Matlab and Netlab it soon become clear that Matlab on a 32-bits operating system has some serious memory issues[65] which limit the amount of training data and number of mixtures. The problem is based on the way 32-bits Windows and Unix addresses and limit the virtual memory. The total amount of virtual memory has a maximum of 2 gigabytes(with a hack this was increased to 3). At first this seems enough for big arrays and matrices, but the problem is that Matlab must store arrays in memory(virtual) in continuous blocks. When this memory gets defragmented by .dll files used by Windows and the largest continuous block is 1024 megabyte, problems arise. To check if this was a problem with the implementation of Netlab, the Data Description Toolbox for Matlab[64] was tried on the same tasks and the same shortcomings were discovered. A test feature set extracted from the BBC Sound Effects Library[53] covering 2,2 hours of sound has 701579 instances(examples) and 40 millions entries. The features were 7 MFCCs and 49 SDCs for each example together with one class label. The training of a GMM with 32 mixtures got the system to kneel with an *out of memory* error. The wishes to explore higher number of mixture models forced an installation of an x64 bits operating system. Then the training of a GMM with the above mentioned feature set' was possible with 64 mixtures. More than this slowed down the training process to almost still-standing because of the swapping to hard-disc. Additional tools used were *PLP and RASTA (and MFCC, and inversion) in Matlab* by Dan Ellis together with the tools needed to be implemented for the task.

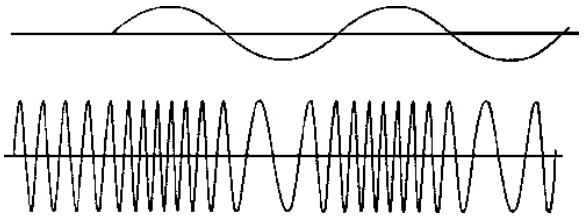

Figure 1.5: The figure shows how the carrier(bottom) varies in frequency as the modulating(above) signal changes in amplitude.

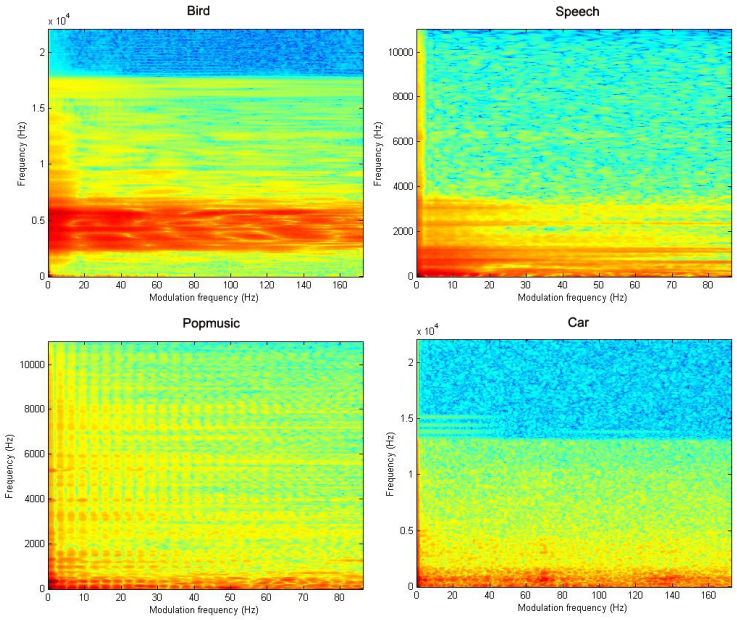

Figure 1.6: [37].Modulation spectrum of speech can be obtained by spectral analysis of temporal trajectory of power spectral component of speech.

1.7 Features

Some machine learning systems as Data description toolbox (dd_tools)[64] which is build on top of and is dependent on The Matlab Toolbox for Pattern Recognition[21] are able to train a one class classifier with target class data but also make use of outliers data. This could in some situations make a more robust classifier. The main problem is still to get a good discrimination between music, speech and other sounds. From literature many known features are known and combinations and subsets of them will be tested.

1.7.1 MFCC

Explained in chapter3.

1.7.2 Spectral Centroid

Spectral Centroid[62] is the balancing point of the spectrum. It can be calculated using :

$$C_t = \frac{\sum_{n=1}^{N} M_t(n) * n}{\sum_{n=1}^{N} M_t(n)} \tag{1.1}$$

where $M_t(n)$ is spectrum of the FFT at t-th frame with a frequency bin f. The Spectral Centroid is a measure of the spectral shape.

1.7.3 Spectral Flux

Spectral Flux[62] is defined as the variation value of spectrum between two adjacent frames :

$$SF = \|N_t(f) - N_{t-1}(f)\| \tag{1.2}$$

Where $N_t(f)$ and $N_{t-1}(f)$ are the normalized magnitude of the FFT at the current frame t and previous frame t-1. Spectral Flux is a a measure of the amount of local spectral changes.

1.7.4 Spectral Rolloff

Spectral Rolloff[62] is the frequency below which 85 percent of spectrum distribution concentrated. It is also a measure of the spectral shaper.

1.7.5 Modulation frequency

Frequency modulation as known from digital signal system, is when the frequency of the transmitted signal varies in proportions of input signal X(t)[34]. The amplitude of the carrier is kept constant and its frequency varied by the modulating signal.See fig.1.5.

There exist substantial evidence that many natural signals can be represented as low frequency modulators which modulate higher frequency carriers[60]. On average, the modulations present in the speech envelope have larger amplitudes at modulation frequencies between 2 and 16Hz, with a dominant peak at 4Hz[23]. The latter is a well known feature and used by[71]. This is a feature worth looking into, and maybe use some statistics of modulation spectrum. The fig.1.6 shows a few examples of modulation spectra. The pictures are made with the ISDL Modulation Toolbox[37]. There is a lot of information in these pictures. First, there is some evidence that speech has most of its energy in modulation frequency 2-8 Hz. One other thing with speech is that most of the energy is below 4kHz in the spectral domain. For music, one can observe that the energy is evenly distributed over the whole modulation domain. In both, the bird example

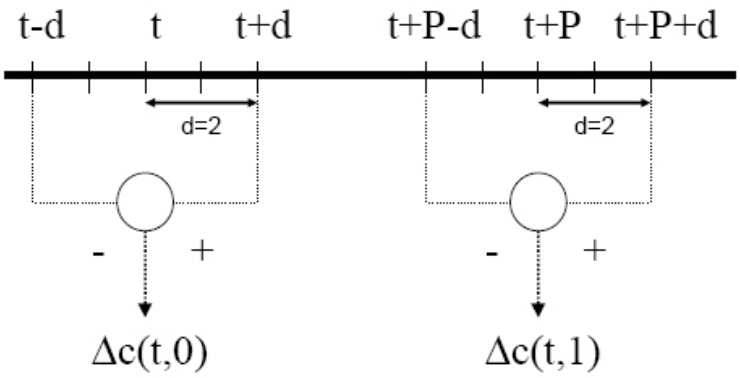

Figure 1.7: [10]SDC computation

and the music example, the energy is evenly distributed over the modulation domain. Also clearly shown in the car example. If one takes a look at the very left side of the frequency axis, the information from modulation frequency 2Hz to 0 is probably garbage data because of the sample length of 0.5 second. This is according to the formula below. Accuracy down to modulation frequency 1Hz require 1 second of data.

$$T = \frac{1}{f} \tag{1.3}$$

1.7.6 SDC

Shifted Delta Cepstral (SDC)[36] are obtained by concatenating the delta-cepstral computed across multiple frames of speech data. This is an extension of the delta and delta delta cepstral coefficients where one gets the ability to capture additional temporal information spanning multiple frames about the speech into the feature vectors.

The SDC features are specified by 4 parameters N, d, P and k, where N is the number of cepstral coefficients, d represents the time advance and delay for the computation of the delta, k is the number of deltas that are concatenated to form the final feature vector, and P is the time shift between consecutive blocks.

N cepstral coefficients are computed for each frame and for each of these blocks of cepstral coefficients the final SDC vector at time t is given by the concatenation of all Δc(t + iP) for all 0 ¡i¡k where

$$\Delta c(t + iP) = c(t + iP + d) - c(t + iP - d) \tag{1.4}$$

See figure.1.7 for a graphical representation of SDC.

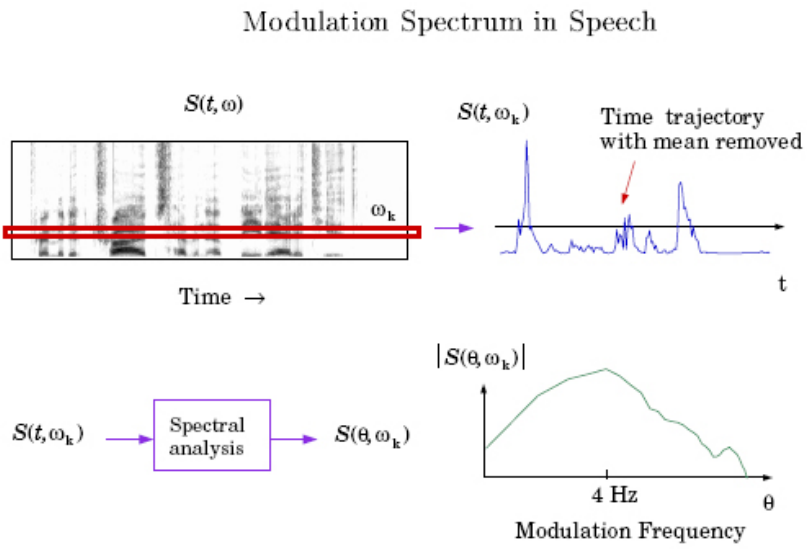

Figure 1.8: [28]Modulation spectrum can be obtained by spectral analysis of temporal trajectory of power spectral components.

1.8 GMM System

The implemented GMM system for music/speech discrimination consists of a feature extractor, two(three) GMM models and a test module, see fig.1.9. The decision part is adjustable so that one can make a decision each frame(typically 20 milliseconds) and up to the whole utterance length. The file *feature_extract2.m* is the feature extractor. The file *test_gmm.m* is the test module while *make_gmm.m* is the training module. The latter is common for both speech/music/other and the LID system. The system was implemented in Matlab.

1.8.1 Experiment 1 - Number Of Iterations For The EM Algorithm

First the number of iterations for the EM-algorithm verses accuracy were tested. The number of iterations for the k-mean algorithm, for setting the initial centers, was set to 500 for all experiments. Trained GMM models with diagonal covariance matrices for speech and music using 12 MFCC(first excluded) were made based on the samples. The number of mixtures for this test was 32 and the number of EM iterations set to 500,250,100,50 and 10. This is done because further testing with a low as possible number of EM iterations, without losing too much accuracy will be preferable. In this test the GMM 32 converged at 462 iterations. The decision length for the test system is set to 20 milliseconds which means that the system must make

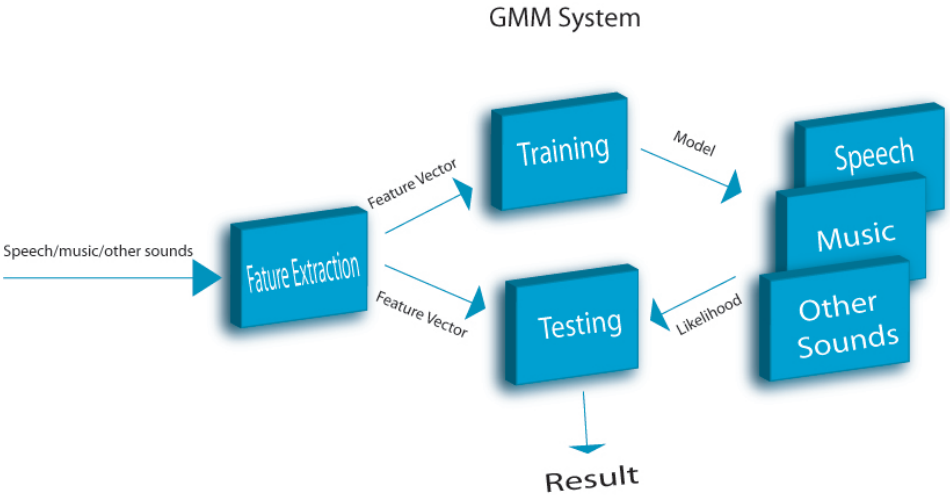

Figure 1.9: Overview of GMM system

iterations	speech error	music error	total
10,00	26,53	14,68	18,57
50,00	25,54	14,41	18,06
100,00	25,51	14,69	18,24
250,00	25,64	14,96	18,46
500,00	25,48	14,20	17,90

Table 1.4: EM iterations

an music/speech decision every 20 milliseconds and all these decisions will count for the error rate. The feature set is extracted from the speech and music corpus. 60 music samples and 60 speech samples were used. This set has it own test set for comparison with other experiments. These will be used for testing the accuracy.

The result of this test is found in table 1.4 and it is clear that the EM algorithm converges quickly to a good state due to it's local search nature. Typically, clustering algorithms, like the k-mean, are initialized by random starts and therefor the differences in the table is probably not significant. The mixture model is initialized using a small number of iterations of the K-mean algorithm. With this dataset it seems like it is not necessary to let the EM algorithm converge.

1.8.2 Experiment 2 - Number Of Mixtures

In this experiment the number of mixtures verses accuracy will be tested on this speech/music corpus. 12 MFCCs(power coefficient excluded) are extracted as features and the covariance matrix is set to diagonal for now.

decision interval	music error	speech error	total error
20 ms	13,59	36,76	21,19
100 ms	11,43	32,15	18,22
500 ms	8,07	21,55	12,49
1000 ms	6,27	15,71	9,37
5000 ms	2,44	10,00	4,92

Table 1.5: 1 mixture, diagonal co-variance matrix

decision interval	music error	speech error	total error
20 ms	16,96	31,61	21,76
100 ms	14,45	26,75	18,48
500 ms	9,50	16,04	11,65
1000 ms	8,18	11,07	9,13
5000 ms	1,22	5,0	2,46

Table 1.6: 2 mixture, diagonal co-variance matrix

The number of iterations for the K-mean and the EM algorithm were both set to 50.

The selection of mixture number K is dependent on the amount of available training data. It needs to be large enough to accurate model the acoustic variability of the speech utterances. On the other hand, it should also be small enough to allow for a reliable estimate of the model parameters[29]

Ideally the number of mixtures should approximate the number of natural classes in data. If the number of mixtures is less than the number of natural classes, then closely placed clusters will fuse to form larger clusters whose variance is higher[5]. This results in under fitting of the data. If the number of mixtures is more than the number of natural classes, larger clusters are broken up into smaller sub-clusters. The new clusters will have lower variance and some of the consequences is : a normal distribution with a low variance has a sharp slope, meaning that the frames that lie close to the mean get a high score but the scores drop fast as the distance increases. This means an increase in average and low scoring frames. This is how over fitting a GMM works.

Table 1.5 to table 1.10 shows results of different numbers of mixtures for the same data. From 1 mixture up to 16 mixtures the accuracy increases in small steps, but from 16 to 32 mixtures a small drop is seen. This could mean that the natural number of clusters in this dataset is between 16 and 32. Another thing to note is how little the accuracy increase from 1 mixtures up to 16. This could be the nature of the dataset.

decision interval	music error	speech error	total error
20 ms	11,95	29,40	17,68
100 ms	9,31	23,32	13,91
500 ms	5,97	12,76	8,20
1000 ms	3,48	9,29	5,39
5000 ms	1,22	5,0	2,46

Table 1.7: 4 mixture, diagonal co-variance matrix

decision interval	music error	speech error	total error
20 ms	11,69	28,15	17,09
100 ms	8,53	22,52	13,11
500 ms	6,13	12,76	8,31
1000 ms	3,31	10,36	5,62
5000 ms	1,22	5,0	2,46

Table 1.8: 8 mixture, diagonal co-variance matrix

decision interval	music error	speech error	total error
20 ms	14,22	26,73	18,31
100 ms	10,02	21,14	13,66
500 ms	6,31	11,72	8,08
1000 ms	4,18	10,36	6,21
5000 ms	0	2,5	0,82

Table 1.9: 16 mixture, diagonal co-variance matrix

decision interval	music error	speech error	total error
20 ms	14,63	26,38	18,49
100 ms	10,92	20,07	13,92
500 ms	7,65	10,52	8,59
1000 ms	5,05	8,21	6,1
5000 ms	1,22	2,5	1,64

Table 1.10: 32 mixture, diagonal co-variance matrix

num MFCC	music error	speech error	total error
4	4,70	13,93	7,73
8	6,27	11,42	7,96
12	4,18	10,36	6,21
14	3,83	7,50	5,04
16	4,18	7,86	5,39
18	3,14	7,86	4,68
20	2,44	7,86	4,20
22	3,30	6,43	4,33
24	2,78	6,43	3,98

Table 1.11: 16 mixture, diagonal co-variance matrix, different number of MFCCs and 1 second decision interval

num MFCC	music error	speech error	total error
4	4,18	13,93	7,37
8	4,70	10,00	6,44
12	5,05	8,21	6,10
14	3,48	6,43	4,45
16	2,61	7,85	4,33
18	2,61	7,14	4,09
20	2,61	6,79	3,90
22	3,14	4,10	4,10

Table 1.12: 32 mixture, diagonal co-variance matrix, different number of MFCCs and 1 second decision interval

1.8.3 Experiment 3 - Number Of MFCCs

In this test the number of mixtures are set to 16 and 32 for an investigation of the number of MFCCs. As seen in table 1.11 the optimal number of mixtures for 16 mixtures is 8 and for 32 mixtures in table 1.12 the number is 4. The conclusion of this is that there exist no such thing as an universal correct number of MFCCs when dealing with GMMs.

1.8.4 Experiment 4 - Adding Delta and Delta Delta MFCC

Delta features is a well known feature used to capture temporal information. Comparing table 1.13 with table 1.9 one see a quite big increase using delta features together with GMMs(and probably Perceptual Linear Predictive (PLP) as well). Comparing table 1.13 and table 1.14 one sees again that the increase to 32 mixtures for this dataset is not making much differences. From table 1.15 it become evident that using delta of deltas in addition to delta coefficients slightly increase the accuracy.

decision interval	music error	speech error	total error
20 ms	7,82	22,87	12,75
100 ms	4,86	17,08	8,80
250 ms	3,51	12,46	5,45
500 ms	2,36	10,17	4,91
1000 ms	1,57	7,14	3,39
3000 ms	0,61	5,00	2,05
5000 ms	0,00	2,50	0,82

Table 1.13: 16 mixture, diagonal co-variance matrix, 12 MFCC and 12 Delta

decision interval	music error	speech error	total error
20 ms	7,94	21,57	12,41
100 ms	4,71	16,11	8,45
250 ms	3,22	11,78	6,03
500 ms	2,02	9,66	4,52
1000 ms	1,05	6,43	2,81
3000 ms	0,00	5,00	1,69
5000 ms	0,00	2,50	0,82

Table 1.14: 32 mixture, diagonal co-variance matrix, 12 MFCC and 12 Delta

decision interval	music error	speech error	total error
20 ms	7,26	22,91	12,39
100 ms	3,90	17,15	8,24
250 ms	2,56	11,95	5,64
500 ms	1,43	10	4,24
1000 ms	0,69	6,07	2,46
3000 ms	0,00	5,00	1,69
5000 ms	0,00	2,50	0,82

Table 1.15: 16 mixture, diagonal co-variance matrix, 12 MFCC,12 Delta and 12 Delta Deltas

decision interval	music error	speech error	total error
20 ms	2,70	26,83	10,61
100 ms	1,31	23,96	8,70
250 ms	0,7	20,67	7,2
500 ms	0,25	16,38	5,54
1000 ms	0,00	12,86	4,22
3000 ms	0,00	7,50	2,46
5000 ms	0,00	2,50	0,82

Table 1.16: 16 mixtures, diagonal co-variance matrix, SDC7-1-3-7

decision interval	music error	speech error	total error
20 ms	2,69	23,47	9,50
100 ms	1,26	20,03	7,42
250 ms	0,7	17,71	6,22
500 ms	0,334	13,27	4,58
1000 ms	0,00	8,57	2,81
3000 ms	0,00	5,00	1,60
5000 ms	0,00	2,50	0,82

Table 1.17: 64 mixtures, diagonal co-variance matrix, SDC7-1-3-7

1.8.5 Experiment 4 - Adding SDC

Shifted Delta Coefficients are an extension to Delta coefficients and will be further explained in chapter 3.

1.8.6 Experiment 5-Testing against other sounds

It could be optimistic to believe that one could discriminate between music/speech and other sounds with these models. The Netlab toolboxes[1] function gmmprob() output probability density values. When making deci-

decision interval	music error	speech error	total error
20 ms	3,00	23,30	9,65
100 ms	1,75	20,00	7,73
250 ms	1,36	17,29	5,59
500 ms	1,09	14,14	5,37
1000 ms	0,87	11,42	4,33
3000 ms	0,61	6,25	2,46
5000 ms	0,00	2,50	0,82

Table 1.18: 16 mixtures, diagonal co-variance matrix, SDC7-1-3-7 + 7 MFCCs

decision interval	music error	speech error	total error
20 ms	3,17	21,04	9,04
100 ms	1,51	17,58	6,78
250 ms	1,03	15,51	5,77
500 ms	0,76	11,72	4,35
1000 ms	0,17	8,93	3,04
3000 ms	0,00	6,26	2,05
5000 ms	0,00	2,50	0,82

Table 1.19: 32 mixtures, diagonal co-variance matrix, SDC7-1-3-7 + 7 MFCCs

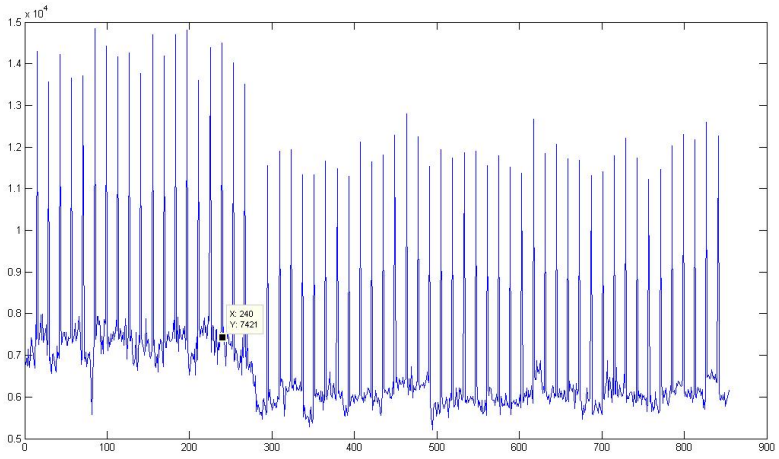

Figure 1.10: -log likelihoods for music model

sions over longer intervals one must add the -log likelihoods. Now, a small probability density value becomes large which means that bigger values are less probable. In fig.1.10 and fig.1.11 the -log likelihoods are recorded every second and one could maybe draw a border for outliers based on these densities. To accomplish a test, 40 random sounds were picked from the BBC Sound Effects Library[53]. The picked sounds range from baby sounds and a river flowing to explosions. The length of the sounds range from 1-3 seconds. The probability densities for 'other sounds' had to be larger than $0,8 - 0,82 * 10 \exp -4$ to be discriminated from speech and music. This was implemented in the test function and the error rate for other sounds were about 90 percent. It was learned that the probability densities for 'other sounds' mix so close to speech and music that these discriminations were impossible using this simple approach.

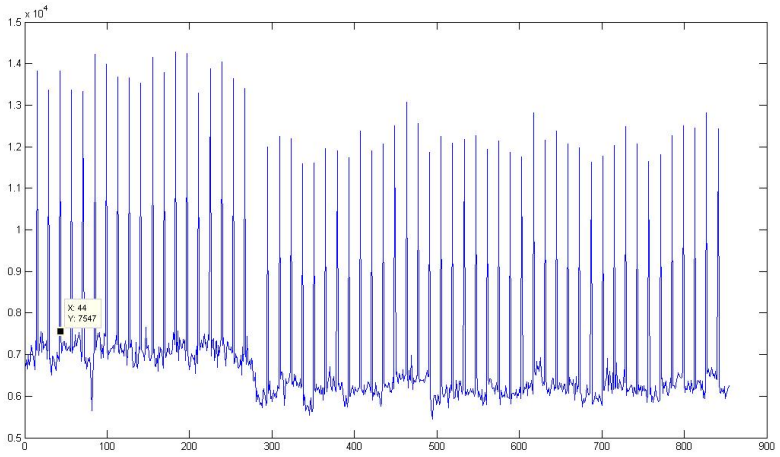

Figure 1.11: -log likelihoods for music model

1.9 Another Approach

When using GMM models it soon became clear that modeling music and speech do not give the ability to discriminate a third class not modeled. Another approach would be to make and model this third class using GMM. For this purpose a lot of 'other sounds' samples had to be used. BBC Sound Effects Library[53], mentioned above, is an enormous database of sound effects and it's main categories are :

- Exterior atmospheres
- Household
- Interior backgrounds
- Transport
- Animals and birds
- Human crowds, children and footsteps
- Comedy, fantasy and humor
- International
- Communications
- Water
- British birds

- Industry
- Cities
- Natural atmospheres
- Cars
- Sport and leisure
- Bang!
- Electronically generated sounds
- Weather
- Ships and boats
- America
- Aircraft
- China
- Babies
- Hospitals
- Africa: the human world
- Africa: the natural world
- Equestrian events
- Greece
- Adventure sports
- Livestock
- Farm machinery
- Horses
- Horses and dogs
- Schools and crowds
- Spain

The length of the sounds range from a few seconds to several minutes and contains over 1500 sounds. The library is delivered on 40 CDs and the total playtime is over 40 hours covering an enormous range of sounds not being speech nor music. The samples were divided into groups of training instances and test instances of ratio 1 3.

Decision Int	Speech Error	Music Error	Other Error	Total Error
0,5 s	11,11	11,92	8,61	10,45
1 s	7,93	6,25	5,63	6,65
3 s	0	4,88	0	1,98

Table 1.20: Final test for music, speech other discrimination

1.9.1 Experiment - Final

The GMM for other sounds were trained with 64 mixtures and 7 PLP-RASTA and SDC-7-1-3-7. The test results with 1 second decision interval is show in table 1.20. The result was quite good and better than one could ever hope for before starting on this. To get an descent even number for test elements, one had to reduce the test instances of other sounds(40) to equal the number of music which has 40 test instances. Several mixes of test instances were also tested without any big differences.

Chapter 2

Language Identification Using Template based Word Spotting

This chapter describe an attempt to do keyword spotting based on whole-word templates. This approach did fail and another approach were tried. However a short description of the approach are listed in this chapter.

2.1 Introduction

The initial idea was to test a whole-word template based word spotting system not based on phoneme recognition. The lack of phonemically transcribed data for multilingual languages was the main reason for not using phoneme dependent systems. The system was going to be able to detect language with the use of word spotting. The idea was to make templates of, let's say 100 most used words for each language and then do a pattern matching algorithm on the input stream with the set of multilingual templates.

2.2 Word Spotting

Word spotting in general is the ability to spot words in continuous speech, in environment where the words are either isolated or connected. The difference is the pauses and the length of them. I continuous speech there is often no pause between the words and in isolated word detection one expects just one word. This could be a command to the radio : 'on' or 'off'. Connected words are words spoken without any considerable pause between them.

The interest for word spotting stated early in the 1970s, and was based on templates. This meant that one had templates or models for words and then

used some sort of pattern matching to check if a word matched the template. The templates could be Linear Predictive Coding of the words as in[14]. A dynamic programming algorithm was used to find the minimum distance between the templates and the incoming words. This experiment had to do with continuous speech and on paper it seem like reasonable results. The speaker dependent tests shows 100 percent accuracy for 4 words, but as seen in the paper different parameters were adjusted for each word for the same speaker. The same is true for the independent test, one had to adjust parameters for different words and speakers. [54] used autocorrelation coefficients to make templates for independent isolated word recognition.

The pattern matching problem was solved by different variations of dynamic programming algorithms called dynamic time warping. The problem these solve is that word utterances differs in time. Here is where the dynamic time warping algorithm comes in handy, it finds an optimal match between two sequences of feature vectors which allows for stretched and compressed sections of the sequence.See [47] for an overview of the different algorithms in this category.

One of the main problems of continuous whole word template matching is illustrated in fig.2.1. Here the word '3' and '8' are spoken in isolation and continuous '38', and one can see that the boundaries are completely erased. The other big problem is co-articulation which means that the pronunciation of a phoneme is dependent on the phonemes preceding and following it[20]. This means that a word articulation changes in different contexts. This followed by the general difference in articulation and the other differences in the speech production system of the human, make this a difficult task.

Today's template matching using whole words as templates is not used much, at least in systems where speaker independence is important. Instead a state of the art Keyword Spotting system uses a speech recognition system based on Hidden Markov Models[70].

2.3 Continuous Dynamic Time Warping

There exist several dynamic time warping algorithms for use with a continuous data stream, and some claim to do real time detection[30, 31]. Common for these algorithms is that they are all based on the traditional dynamic time warping. The dynamic time warping algorithm of[49] was implemented in Matlab for testing purposes. This algorithm gives a local minimum(under a threshold) value when a template matches the stream. In this domain, as with other speech applications it is important to reduce the data because raw speech data is too much to handle. As with other speech recognition/application tasks the Mel-frequency cepstral coefficients (MFCCs) and the perceptual linear prediction (PLP) coefficients is a natural choice of data reduction tools.

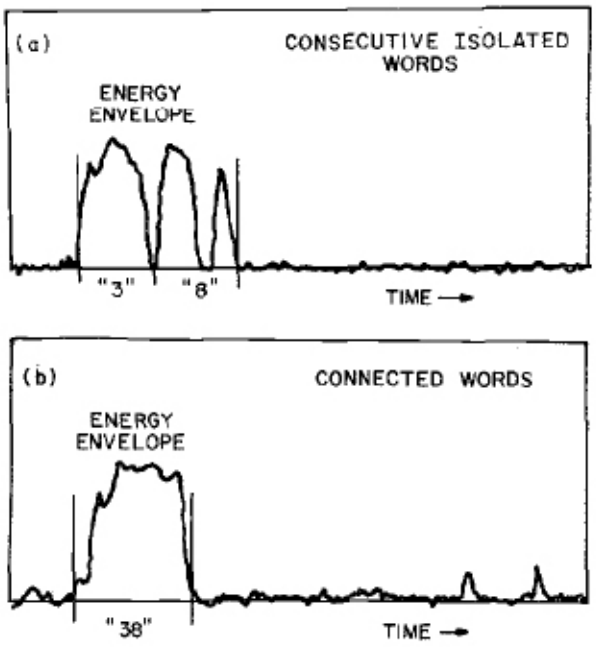

Figure 2.1: [46] Log energy for two speech utterances.

2.4 Pre-Study

The TIMIT corpus[22] consist of phonemically and lexically transcribed speech of 630 American English speakers from 8 major dialect regions (labeled from DR1 to DR8). The speech signal is recorded through a high quality microphone. All tests contain the same dialect(DR6) New York. As a test, one speaker was chosen, FAPB0 a woman, and different words were extracted from the sentences according to the transcribed data.Fig.2.2 shows the word 'awkward' spoken by the same speaker. One normally spoken and one with the pressure on the beginning of the word. In fig2.3 a plot of the 13 first MFCC for both utterances of the word 'awkward'. One sentence containing both 'awkward' was set as a stream and one of the extracted utterances was set as template. The result of this run is in fig.2.4. Both 'awkward' were a local minimum even though the waveforms and the MFCC plot were so different. The result is not so good in despite of the sentence used : 'It was awkward: very awkward.'(SI1693). Next the sentence, 'Don't ask me to carry an oily rag like that(SA2)' spoken by the same speaker FAPBO and the word 'carry' extracted from another speaker(FBCH0), also a woman. The result of the run with the CDP-algorithm is shown in fig.2.5. Here the correct point is not the local minimum with the lowest value, but it's not far away. Another test using the same sentence is seen in fig.2.6 between FAPB0 and FKLC1, where FKLC1 extracted the word 'carry' is

the reference and FAPB0 is the stream. Here the match is rather clear. This test was done with 8 MFCCs. Doing tests with various MFCCs showed little to no significant differences between 8 and 20 MFCCs. Also did a small test with MFCCs and theirs delta MFCC without any substantial differences besides more noise.

This fast check did not give an answer if the speaker independent recognition rate is good enough for quality word spotting. The speaker independent word spotting is difficult to check in a big scope because there are not many tests per speaker one can do. The words that are common for the same speaker are words like : her, is, the and in. To cut out words from a sentence for checking for a speaker will be a waste of time. The MFCCs wont be identical, if we are not so lucky to hit the exact point where the last coefficient in a frame ends, and also have the same luck at the end. If, not the MFCCs wont be identical but very near identical and a match should be an easy task. Besides the speaker dependent recognition rate is of little or no interests in the application tested here.

2.4.1 Testing Speaker Independent Spotting

One simple way of thinking is that if one sentence contains one reference word, this word should be the one with the lowest value for the local minimum. In the same way one can say that if the sentence contains two instances of the word, then they should be the two with the lowest local minimum. This is what we know for this test. In real life it could be that a sentence does not contain a reference word but the path of the algorithm would still have several minimum points, and in these situations one should have threshold values connected to each template. In this way one can conclude that an imaginary, say 'house' can not be that minimum because it's threshold is below the minimum point. The TIMIT corpus contains several sentences from several people, both men and women, and from different parts of USA. A Matlab script was made to extract the words from the sentences according to the transcription which says where a word begins and ends. To limit the test, only two sentences will be used : SA1 and SA2, which contain the sentences : 'She had your dark suit in greasy wash water all year.' and 'Don't ask me to carry an oily rag like that. Several speakers will be selected, both men and women and with different dialects. Then a systematic test will be made manually to determine if a match occurs or not. A speaker will sometimes be the stream, where the whole sentence is used, and sometimes a reference where one word is extracted and tested against other streams(sentences). 13 MFCCs were used for the test and 8 speakers, 4 women and 4 men were selected from each district. A total of 5 dialects were used. The total matching accuracy was slightly under 60 percent. For women in the same district the average match accuracy was 67 percent and 65 for men. In this test one knew that the words were present

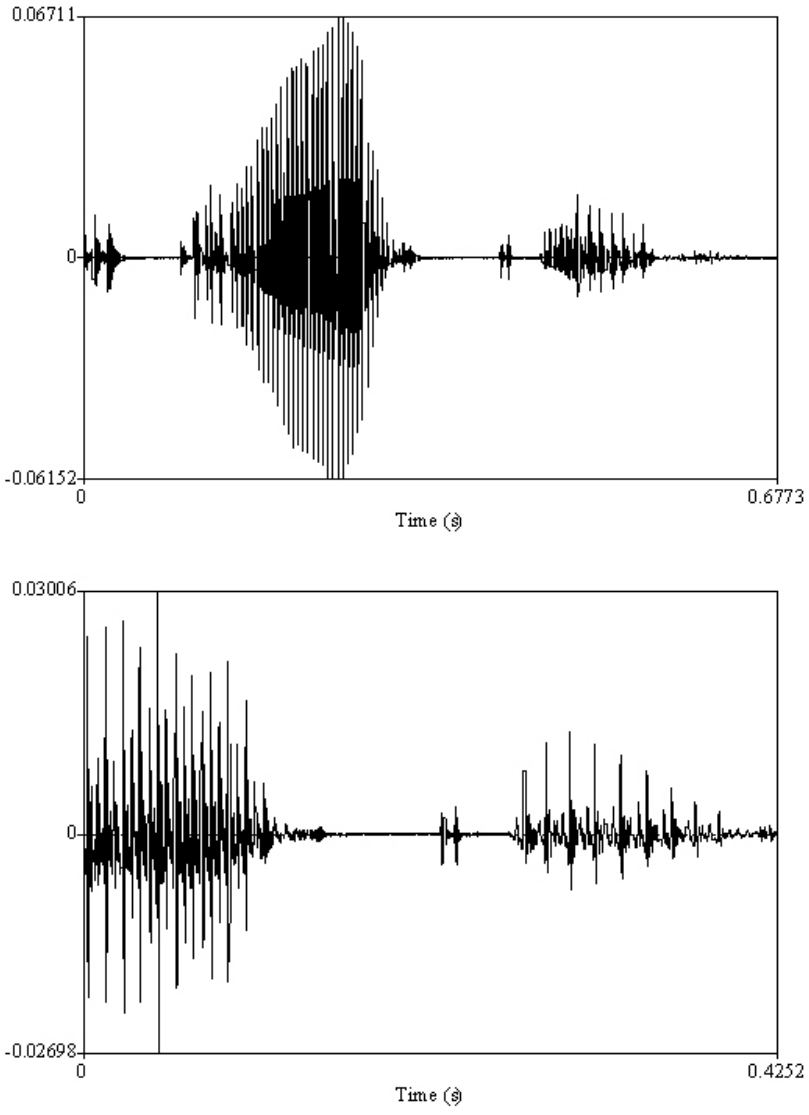

Figure 2.2: The word awkward spoken by the same speaker.

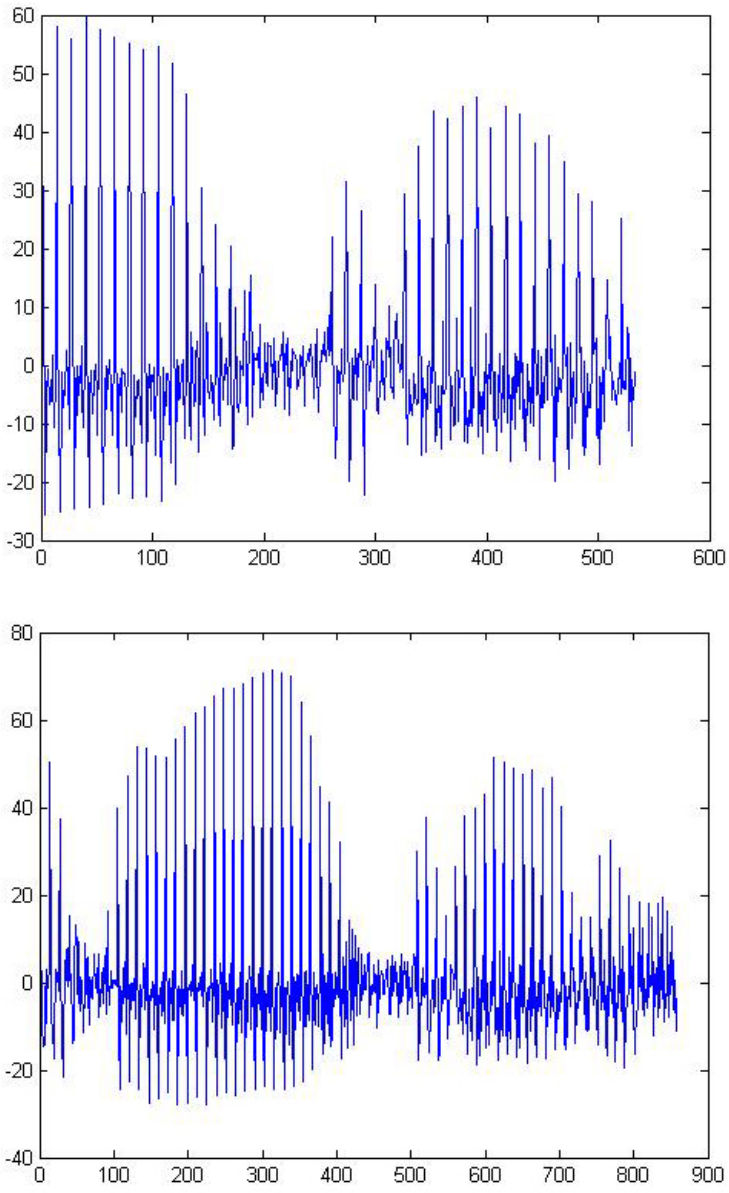

Figure 2.3: Plot of the MFCCs of the word awkward spoken by the same speaker

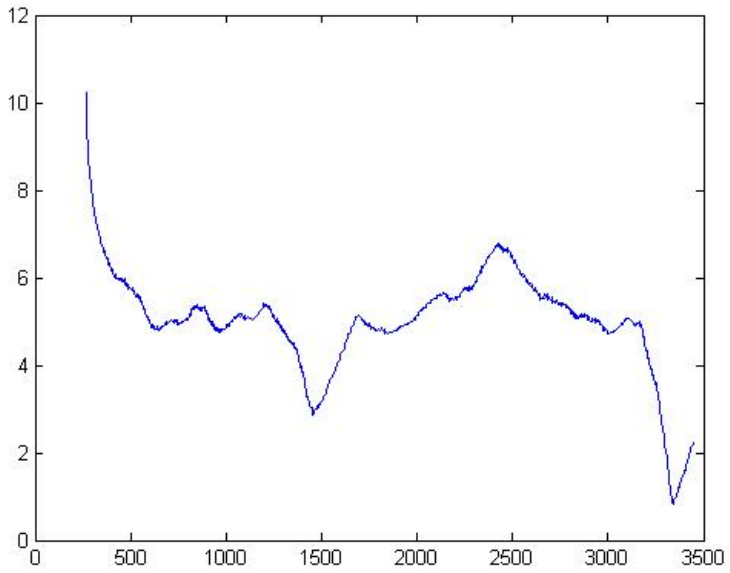

Figure 2.4: Local minimum points indication a match

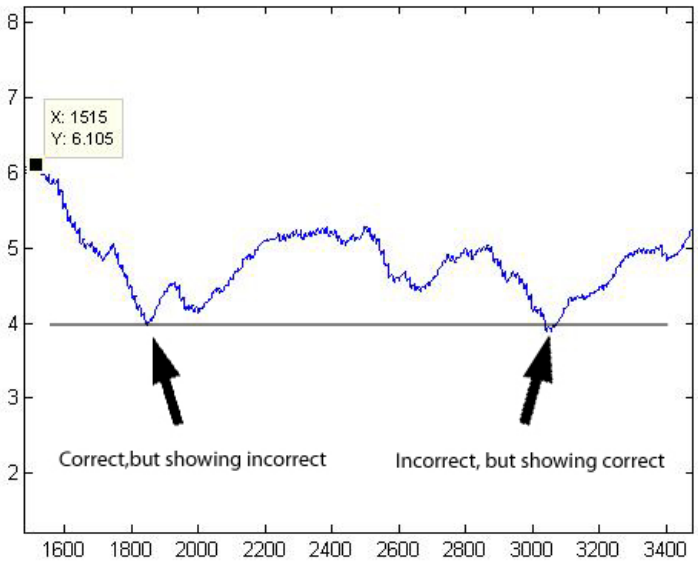

Figure 2.5: Local minimum points indication an incorrect match

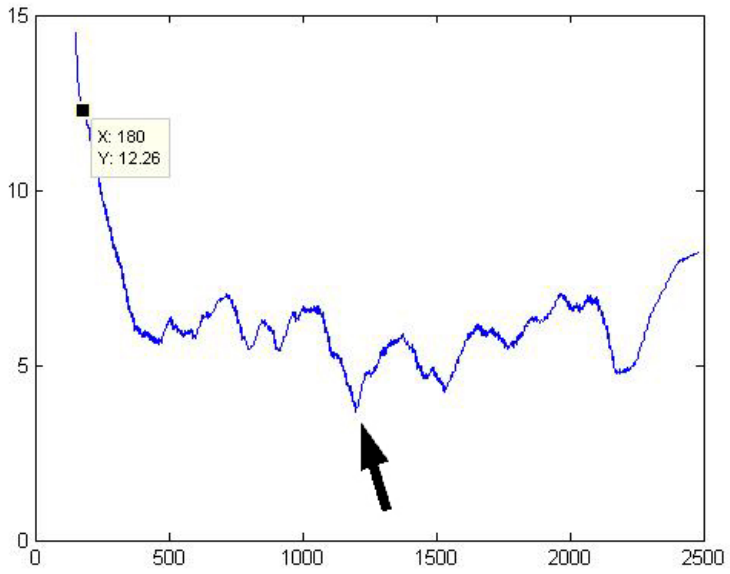

Figure 2.6: Algorithm indication a match in a speaker independent test

in the sentence so the only task was to locate the minimum of the graph from the algorithm. Then one had to determine if the words location in the sentence matched the minimum. Sounds like an easy task, but when one don't know if the word is present or not it could lead to massive amount of false matches even with thresholds attended to each template. To determine the thresholds for the templates one could run the continuous dynamic programming algorithms algorithms against each keyword of the same word. To determine the threshold for a word x one run all other words x against this one and the highest value of the algorithm(the word of the same kind most different from from the word in question).

Chapter 3

GMM based automatic LID system

In this chapter a GMM based Language Identification system is developed based on acoustic features.

3.1 Introduction

Automatic Language identification is the task of identifying a spoken language from a sample of speech[61]

The importance of Automatic Language Identification can be understood by thinking of speech recognition in a multilingual community. A multilingual speech recognizer first have to determine what language to recognize. Other applications are multimedia systems which serve users speaking different languages. This could be a voice controlled service in an international airport or hotel. Routing of telephone calls, especially important for emergency institutions, so that one are switched over to an operator speaking the language in question.

The algorithms for LID can be roughly divided into two groups[75], acoustic and phonotactic systems. In a phonotactic system, a tokenizer transcribes the input speech into phonemes and the decision making is done on strings of phonemes. This approach is called PRLM (Phoneme recognizer followed by language model) or PPRLM (Parallel PRLM)[66][38]. A PRLM language identification system use a large sets of phonemically labeled data for training, and usually one set for each language to be identified. While these systems are effective, the labeling can be time consuming. This means a lot of effort and work extending the system to include other languages[75].

In an acoustic system, the input features are modeled directly by GMMs, neural networks or SVMs and are easily implemented without thorough knowledge of the languages to discriminate. Note that the traditionally kernels, like RBF is not a good choice for classifying sequences like speech

data[9]. The main issue is the inability for SVM for handle feature vectors of unequal lengths. A successful system for language identification or speaker recognition also have to handle sequences of speech data. This means, among other things, the ability to capture temporal information. The sequence kernels developed for these applications implements kernels that compare sequences of feature vectors[10].

3.2 GMM

Gaussian mixture models (GMMs) is considered as a mature semi parametric statistical model and it is widely used to model complex distributions[74]. Usually the parameters of the GMMs are determined in a maximum likelihood (ML) using the Expectation Maximization (EM) algorithm.

Gaussian Mixture Models have been used for a number of applications in the data mining and machine learning field, like time series[39], image detection[24], speaker verification and recognition[55][43] and language identification[4]. The GMM model is a semi parametric model [74].

The EM algorithm is an efficient iterative procedure to compute the Maximum Likelihood (ML) estimate in the presence of missing or hidden data. In ML estimation the goal is to estimate the model parameter(s) for which the observed data are the most likely.

Details of the EM algorithm is found in [42].

Gaussian mixture model :

$$p(o_t \mid C_j) = p(o_t \mid \lambda_j) = \sum_{i=1}^{I} c_i N(o_t; \mu_i, \sum_i)[13] \tag{3.1}$$

Where μ_i is the GMM for class C_j and c_i is a mixture weight that must satisfy the constraint $\sum_{i=1}^{I} c_i = 1$ and I is the number of mixture components. $N(\bullet)$ is

$$N(o_t; \mu_i, R_i) = \frac{1}{(2\pi)^{\frac{d}{2}} \mid \sum_i \mid^{\frac{1}{2}}} {}^{-\frac{1}{2}(o_t-\mu_i)^T \sum_i^{-1}(o_t-u_i)} [13] \tag{3.2}$$

From the pictures 3.1 3.2 and 3.3 one could conclude that full covariance matrices fit the data best, but they are costly in high dimensional feature spaces. Also the use of a GMM with full covariance matrices leads to a huge number of parameters and presents the risk of over-fitting[40][44]. This is strongly dependent of the number of parameters compared to instances in the training set.

Diagonal covariance GMM seems to be a good compromise between quality and model size.

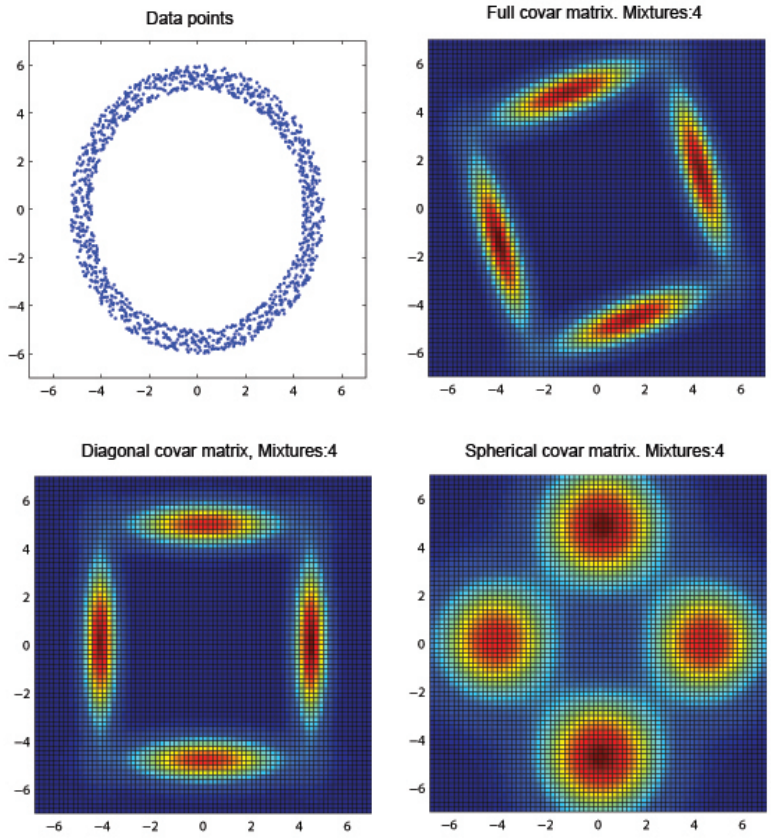

Figure 3.1: Example of the coverage of a GMM model with 4 mixtures from 1256 data points

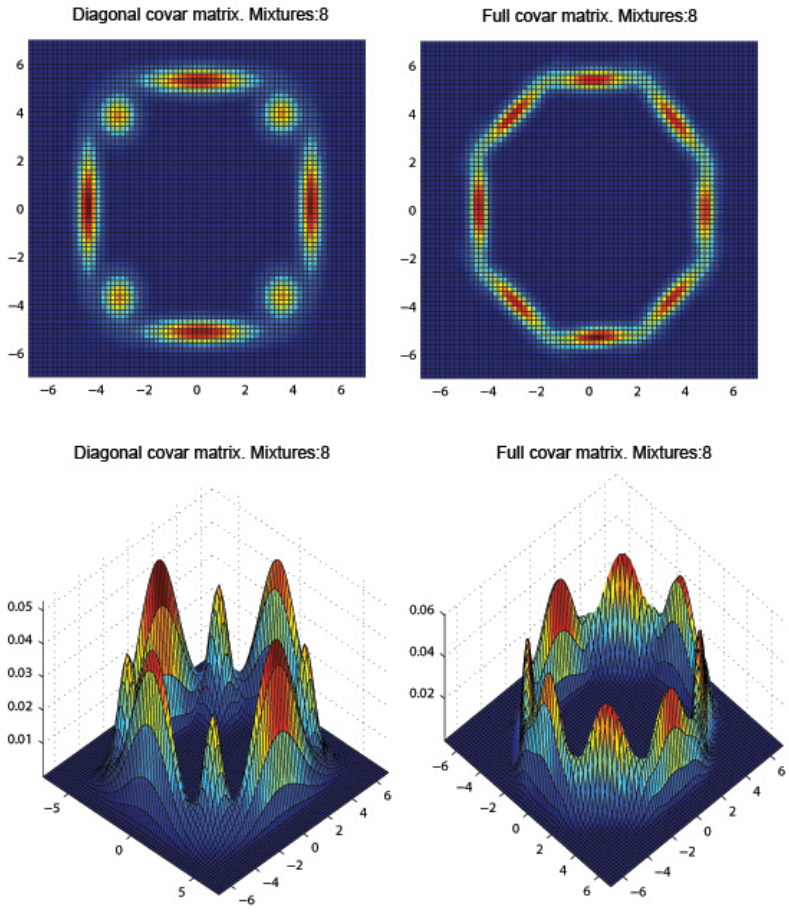

Figure 3.2: Diagonal and full covariance matrices and 8 mixtures

A last Spherical covariance GMM needs many components to cover data, especially in high-dimensional feature spaces. In general,as seen in[44], a GMM with a spherical covariance matrix is too constraint model the data with a high degree of accuracy.

3.2.1 Features

The most common features for speech recognition, speaker recognition and language identification is the MFCCs and PLPCCs. These are the ones that are being the main focus here in one or another form.

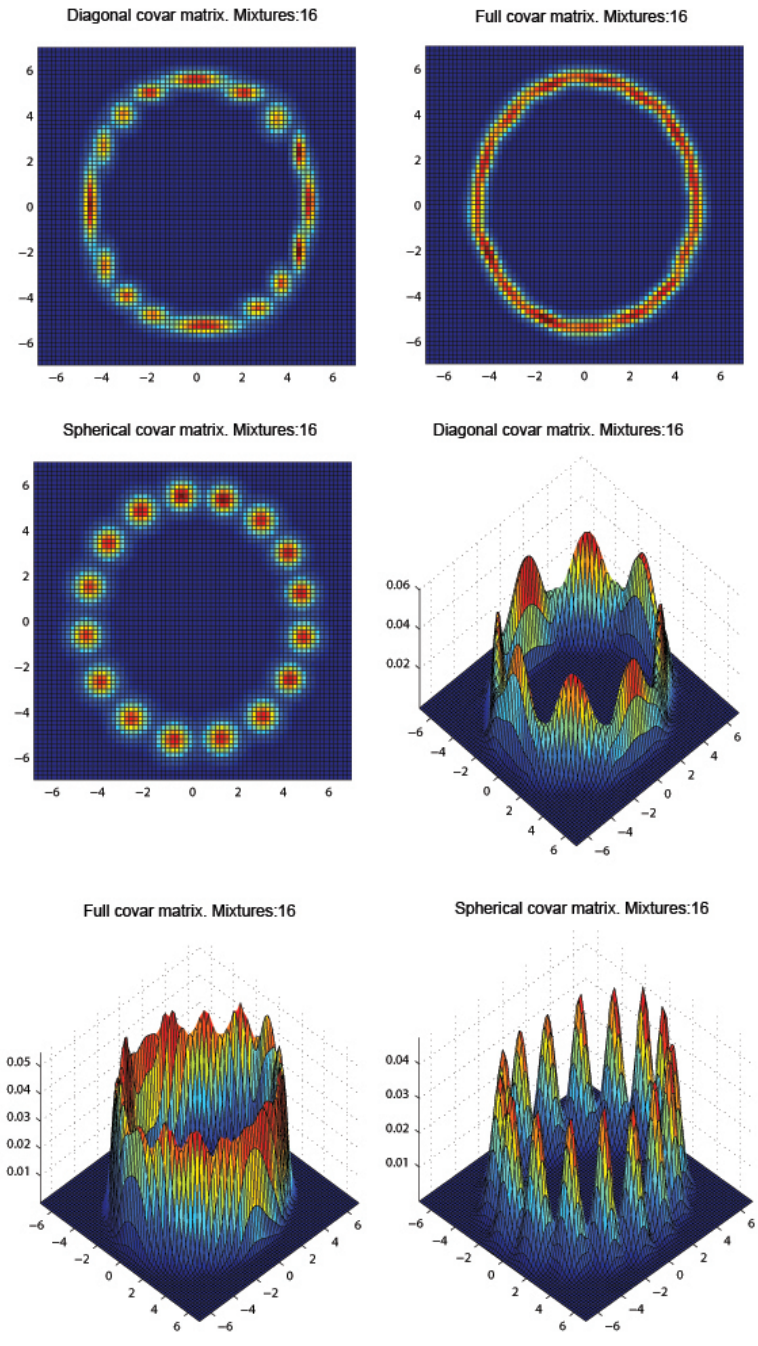

Figure 3.3: Diagonal,full and spherical covariance matrices and 16 mixtures

3.3 MFCC

Psychophysical studies have shown that human perception of the frequency contents of sounds for speech signals does not follow a linear scale. Thus for each tone with an actual frequency, f, measured in Hz, a subjective pitch is measured on a scale called the 'mel' scale. The mel-frequency scale is a linear frequency spacing below 1000 Hz and a logarithmic spacing above 1000 Hz. Fig.3.4 shows an example made with the Audio Processing Toolbox[32] and Matlab. MFCCs are cepstrum coefficients in the mel-scale.It is derived as follows[2]:

1. Take the Fourier transform of (a windowed excerpt of) a signal.
2. Map the powers of the spectrum obtained above onto the mel scale, using triangular overlapping windows.
3. Take the logs of the powers at each of the mel frequencies.
4. Take the discrete cosine transform of the list of mel log powers, as if it were a signal.
5. The MFCCs are the amplitudes of the resulting spectrum.

3.4 Perceptual Linear Prediction

PLPCCs(and also MFCCs) applies greater weight to perceptually-important portions of the spectrum by to modeling some aspects of human perception and methods motivated by the behavior of the human auditory system. The original article is found in[25] along with detailed steps of computation. PLP is often used in combination with RASTA[27].

3.5 Speech Corpus

A subset of the The Linguistic Data Consortiums CALLFRIEND corpus, which is a telephone speech database, is used for testing. The corpus comprises two-speaker, unprompted, conversational speech messages between friends. Hundred North-American long-distance telephone conversations are recorded in each of twelve languages (the same as 11 languages as OGI-TS plus Arabic). There are three sets in this corpus including training, development and test set, each set consists of 20 two-sided conversations from each language, approximately 30 minutes long[15].

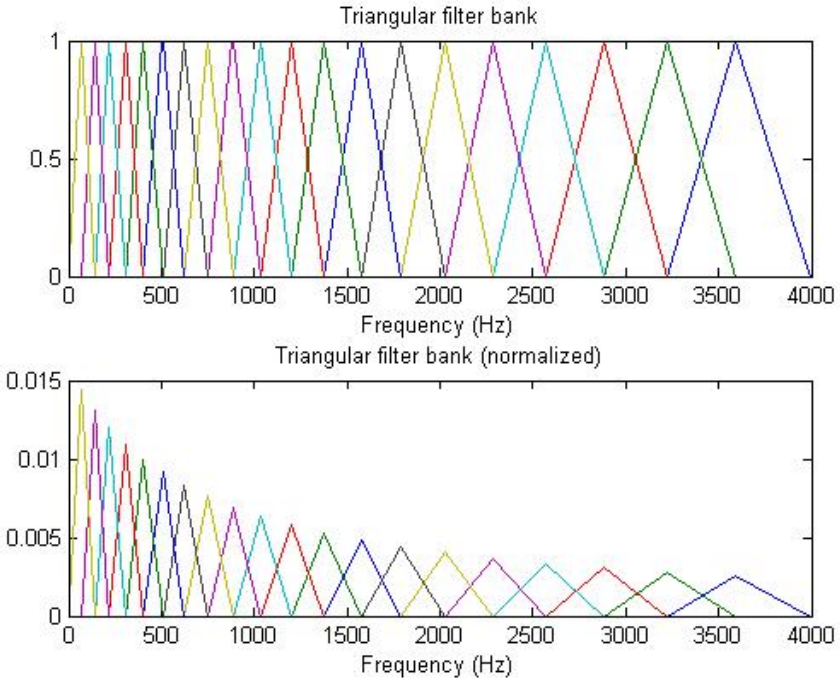

Figure 3.4: Mel filterbanks

3.6 Distortions and Noise

In speech and speaker recognition applications it has been shown that the auditory environment plays a important role. A system trained on noise and distortions free speech data, lose performance in an environment with noise or with channel distortions. The same is true the other way. Most speech recognition and speaker recognition systems use log-magnitude spectrum or any of it's linearly transformed versions like cepstrum as features and the problem is that these are not robust to noise. A channel distortion gives an additive component in the logarithmic spectrum of speech.Any metric based on a short-term logarithmic spectrum (or cepstrum) of speech will reflect this distortion.

3.6.1 Channel Distortion

Channel distortion is caused by a change in the spectral shape of the signal due to the frequency response of the acoustic transmission channel[50]. Some sources of channel distortion are as follows:

- Analog transmission channel. This situation occurs for example in analog telephony, where different subscriber loops have different fre-

quency responses.

- Microphone characteristics. The frequency response of the microphone used to acquire the speech signal is another source of channel distortion, as different types of microphones have different frequency responses.

Any convolutive distortion affects the DC component of the modulation spectrum[48]. This is why methods like mean subtraction on the cepstrum(cepstral mean subtraction) has shown to make cepstral features robust to channel distortions. This method removes the DC component of the modulation spectrum. That efficiency of this method can bee seem in[35][12].

RASTA(RelAtive SpecTrAl) processing is another method that has shown to be robust to channel distortions[26]. It is based on the fact that most channel distortions are slowly changing events, and the method band passes spectral parameter signals to eliminate steady or slowly varying components in the speech signal. RASTA and PLP are often combined to archive robust features[27].

A comparison of RASTA and CMS can be found in [16] and it shows that CMS performs slightly better than RASTA. One explanation of this could be that RASTA always uses the same filter while the filter response for phones varies.

Feature warping[51] is another technique for dealing with noise and channel distortions. It's robustness has been shown in both speaker verification and language detection. The simple description is that feature warping maps the short-term distribution of each feature stream to a standardized distribution, usually a gaussian. In [4] it is also discovered that the normal distribution of the feature stream are very beneficial for an back end GMM because of it's distribution. In[4, 51] it is also shown that feature warping is superior over CMS. Fig.3.5 shows an example of a noisy sound before and after feature warping. The feature warping is done with an implemented algorithm and the histograms are made in Matlab with the *hist* command.

PLP and RASTA (and MFCC, and inversion) toolbox[19] has a build in pre-emphasis filter which is a a one coefficient digital filter :

$$H_{pre}(z) = 1 + a_{pre} * z^{-1} \tag{3.3}$$

Where that range of a_{pre} is in the range [-1,-0,4][52].

Voiced speech naturally have a negative attenuation of approximately 20 dB per decade due to physiological characteristics of the speech production system. The intension of the pre-emphasis filter is to enhance the high frequencies and attenuate the low frequencies.

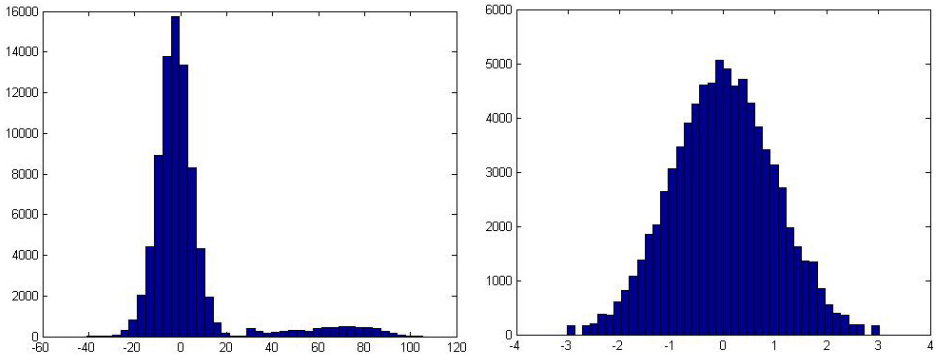

Figure 3.5: Feature warping of noisy speech. Showing before and after warping to normal distribution

3.6.2 Noise

When dealing with telephone as a medium several drawbacks are known, like limited bandwidth, channel distortions as mentioned above, different frequency responses in the microphones, burst noise, echo, crosstalk, envelope delay and clipping[69]. There also exist white noise due to electron agitation in electronic components[63] To compensate for the additive ambient noises of telephone speech one could consider the classical method spectral subtraction[7]. This method often works well on stationary noise like white noise but the drawback is that this method need a 'silence' segment with only the noise to work.

It is usually assumed that the speech and the noise are additive and uncorrelated. The method is simply explained as estimating the noise spectrum in a silence interval and subtract this noise spectrum from the signal spectrum.

Fig 3.6 shows a random pick from the speech corpus to check SNR, this i a silence part,and it looks like white noise[50]. This was the worst example discovered, but it seems like most of the samples contains a considerable amount of electrical noise. The components needed to compute SNR was measured using Praat[6] and calculated according to :

$$SNR = 10 * log(P_{signal}/P_{noise}) \tag{3.4}$$

Where P is maximum pressure. See the documentation of Praat for details.

3.7 Language Identification System

The language system is composed of a feature extractor, a test module and a training module as seen in 3.7.However, this is the schematic view, in the

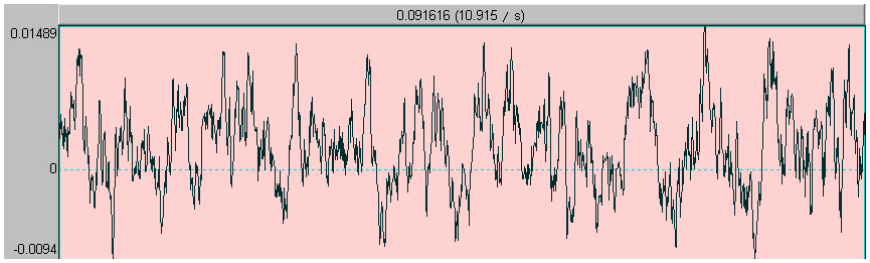

Figure 3.6: A 'silence' part of a taiwan conversation.Giving a SNR=30 db

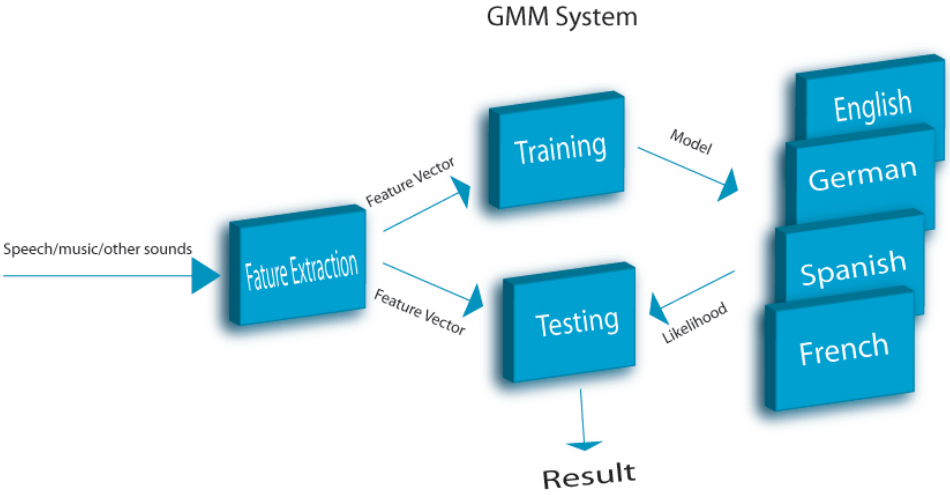

Figure 3.7: Schematic view of the LID system

Matlab code, the test module module has a build in feature extractor. The filenames do not fit this description. The file *feature_extract3.m* is the feature extraction module and *test_gmm2.m* is the test module. The training module is named *make_gmm.m* and is used by both the speech/music/other and the LID system. However, some parameters have to be changed.

3.8 Experiment - MFCC verses MFCC with removed silence

As an experiment 12 MFCCs is used and only 2 languages, German and English.Approximately 20 minutes of speech data is used for each language. A GMM model with diagonal co-variance matrix and 32 mixtures is trained for each language. These tests are not meant to find the optimal numbers of mixtures but rather find the differences between no silence removal and silence removal. It is believed the test results are possible to extend to other features and number of mixtures. If silence removal is the most efficient

Decision Interval	Error English	Error German	Total Error
1 seconds	34,67	31,04	32,76
2 seconds	31,75	25	28,2
3 seconds	29,01	23,89	26,32

Table 3.1: 2 language test, 12 MFCC and 32 mixtures

Decision Interval	Error English	Error German	Total Error
1 seconds	31,22	27,76	29,4
2 seconds	30,95	22,85	26,69
3 seconds	30,24	21,67	25,73

Table 3.2: 2 language test, 12 MFCC,32 mixtures and silence removal

here it is believed that the same is true for other number of mixtures and features like PLP and LPC. For this tests the results are not expected to be very good, but it is the differences that is the main importance here. The amount of training data is considered as a minimum.

Table 3.1 shows the result of language recognition of English and German using 12 MFCC and no silence removal and table 3.2 is the result of removing silence segments from speech before training. The results show that there is a difference but it is not very big and it is suspected that a part of this difference is the fact that the model gets more training data per time unit with silence removal compared to no silence removal. On the other side the only interesting part of speech data is the speech and not the speech, and one can see if the GMMs get 'confused' with all the silence segments between sentences and words.

3.9 Experiment - MFCC,Feature Warping and SDC

As mentioned, feature warping has shown to be very efficient to additive noise and channel effects. That this method really works is evident from the table 3.3 where a interval of 3 seconds is used for the feature warping algorithm. Also a few others intervals were tested without any increase nor decrease of any significant meaning. From literature it is evident that the 3 second interval is, as far as anybody knows, near optimal as seen in [3][8].

On the other hand, what really is quite a surprise is the performance of the SDC, which is not very good, at least in combination with feature warping. As seen in table 3.3 and table 3.4 the effect of adding SDC together with feature warping has an increase that may or may not be significant. The training of an GMM is very dependent of the initial start conditions because of the local search nature of the EM-algorithm. In Netlab, this accomplished with iterations of the k-mean algorithm which has a random start conditions. This mean that training 2 models on the same training

Decision Interval	Error English	Error German	Total Error
0,5 sec	33,05	31,7	32,34
1 sec	25,31	26,11	25,73
2 sec	25,79	25,71	25,75
3 sec	26,54	26,11	26,31
4 sec	23,02	20	21,43
5 sec	26,67	21	23,68

Table 3.3: 2 language test, 12 MFCC,32 mixtures,silence removal and Feature Warping with 3s window

Decision Interval	Error English	Error German	Total Error
1 sec	34,48	31,55	32,94
3 sec	29,01	23,89	26,31
5 sec	25,56	20	22,63

Table 3.4: 2 language test, 8 MFCCs, feature warping and SDC7-1-3-7,32 mixtures and silence removal

data could and probably will give different results.

3.10 Experiment - PLP-CCs

PLPCCs is doing very well as feature set for this task. As seen in table 3.5, 8 PLP-CCs is performing better than 8 MFCCs with feature warping and SDCs in table 3.4. It also beat 12 MFCC and Feature Warping for the 3 seconds task as seen in table 3.3. In combination with RASTA and feature warping PLP-CCs performs even better as seen in table 3.6 and table.3.7.At last it is indeed evident that SDC has an effect, at least in combination with RASTA filtered PLP-CCs as seen in table 3.7. No other combination gives better performance than 7 PLP-CCs and SDC with parameter values 7,1,3,7.

3.10.1 Comparison

For both MFCCs and PLP-CCs increasing the number of coefficients did not make any increase in performance. However, too many coefficients de-

Decision Interval	Error English	Error German	Total Error
1 sec	35,44	27,59	31,31
2 sec	32,14	26,07	28,95
3 sec	30,25	20,56	25,15

Table 3.5: 2 language test, 8 PLP-CC,32 mixtures and silence removal

Decision Interval	Error English	Error German	Total Error
1 sec	34,87	28,97	31,76
2 sec	30,95	24,29	27,44
3 sec	28,4	18,89	23,39
4 sec	26,19	14,29	19,92
5 sec	26,67	15	20,53

Table 3.6: 2 language test, 8 PLP-RASTA,32 mixtures,silence removal

Decision Interval	Error English	Error German	Total Error
1 sec	34,1	27,93	30,85
2 sec	29,37	23,57	26,31
3 sec	29,63	19,44	24,27
4 sec	25,4	13,57	19,17
5 sec	25,56	12	18,41

Table 3.7: 2 language test, 8 PLP-RASTA and Feature Warping with 3s window,32 mixtures,silence removal

decision Interval	Error English	Error German	Total Error
1 sec	34,48	24,67	29,31
3 sec	25,31	15	19,88
5 sec	24,44	12	17,89

Table 3.8: 2 language test, 7 PLP-RASTA and SDC7-1-3-7,32 mixtures and silence removal

creased the performance. With these experiments it was revealed that the PLP-CCs combined with SDCs were superior over other features and combinations. If one check other works done in this field and in the field of speaker recognition and verification there are no true solution what features regards. Some gets nice results with MFCCs in combination with SDCs while others achieve the best results with PLP-CCs. In [3] there is a comparison of PLP-CC and MFCCs in combination with SDCs and feature warping. Here performance of the PLPCCs and MFCCs with different combinations with SDCs and feature warping are quite even, but MFCCs with feature warping and SDCs is slightly betters.[4] only uses MFCCs and they got their best results with MFCC in combination with feature warping and SDCs. In [73] the performance of both PLP-CCs and MFCCs are very even, but here PLP performs better. It is impossible to say that PLP-CCs is better than MCFFs or vice versa. The results varies from a lot of papers and this probably mean that the datasets and their distribution is an important factor.

3.10.2 Experiment - Final

Only a subsection of the whole CallFriend were available. Two dialects of English and Spanish, one from Germany,France and Japanese. A GMM of 64 mixtures were made for English, Deutsch and France, Spanish and Japanese, a total of approximately 1 hour of speech for each model was used. The features used were 7 PLP-CCs-RASTA, and SDC-7-1-3-7. This was the best option for the test runs. However, the test results showed to be no good. Several combination were tried but nothing seemed to reduce the error of especially France which often was taken for both English and Deutsch. The results are shown in table 3.9 for decision intervals 3 and 5 seconds. Confusion matrices are showed in table 3.10 and 3.11.

As mentioned earlier there was some memory problems when training models with a high number of mixtures and data. Also when going over 64 mixtures the covariance matrices had a tendency to collapse during training. This is why there is no results showing more than 64 mixtures, but it was succeeded to get one 96 mixture model for one language : Spanish. The results for using this together with 64 mixture matrices is showed in table 3.12 and 3.13. This results shows that the error for Spanish decreases and i could be a sign for not using enough mixtures for this training data because it is evident that this model models the data with a higher accuracy. But, it is not possible to be sure that increasing all mixtures to 96 or higher would decrease the total accuracy(but probably). The error for Spanish using 3s decision interval was 6,7 percent.

Table 3.14 and 3.15 shows the results of using 12 MFCCs and feature warping and 3s decision interval. This shows that the chosen features performs the best.

Table 3.16 and 3.17 shows the results for the 5 language test with a

Decision Interval	Spanish Error	English Error	Deutsch Error	Total Error
3 Sec	28,75	40,28	25	31,04
5 sec	20	33,33	20	24,14

Table 3.9: 3 language test, 7 PLP-RASTA and SDC7-1-3-7,64 mixtures and silence removal

	Eng	Ger	Spa
Eng	43	17	12
Ger	3	60	17
Spa	14	9	57

Table 3.10: 3 language test, Confusion matrix for 3s test

decision interval of 3s. At last table 3.18 and 3.19 shows the results for the 5 language test with a decision interval of 5s.

	Eng	Ger	Spa
Eng	24	7	5
Ger	1	32	7
Spa	4	4	32

Table 3.11: 3 language test, Confusion matrix for 5s test

Decision Interval	Spanish Error	English Error	Deutsch Error	Total Error
5 sec	0	47,22	50	31,89

Table 3.12: 3 language test, mixtures for Spanish increased to 96

	Eng	Ger	Spa
Eng	19	4	13
Ger	1	20	19
Spa	0	0	40

Table 3.13: 3 language test, confusion matrix : mixtures for Spanish increased to 96

Decision Interval	Spanish Error	English Error	Deutsch Error	Total Error
3 Sec	32,5	33,33	50	38,79

Table 3.14: 3 language test, 12 MFCCs and feature warping

	Eng	Ger	Spa
Eng	24	8	4
Ger	6	20	14
Spa	6	7	27

Table 3.15: 3 language test, confusion matrix : 12 MFCCs and feature warping

Decision Int	Spa Err	Eng Err	Ger Err	Jap Err	Fra Err
3 Sec	40	46,3	40	51,67	56,67

Table 3.16: 5 language test, 7-PLP-RASTA and SDC7-1-3-7, 3s decision interval

	Eng	Ger	Spa	Jap	Fra
Eng	29	10	5	3	7
Ger	2	36	6	0	0
Spa	6	4	36	7	7
Jap	5	7	16	29	3
Fra	8	12	12	2	26

Table 3.17: 5 language test confusion matrix: 7-PLP-RASTA and SDC7-1-3-7, 3s decision interval

Decision Int	Spa Err	Eng Err	Ger Err	Jap Err	Fra Err
5 Sec	14	22,22	30	35	40

Table 3.18: 5 language test confusion matrix: 7-PLP-RASTA and SDC7-1-3-7, 5s decision interval

	Eng	Ger	Spa	Jap	Fra
Eng	14	2	2	0	0
Ger	1	14	0	0	0
Spa	0	0	17	1	2
Jap	0	1	5	13	1
Fra	2	1	5	0	12

Table 3.19: 5 language test confusion matrix: 7-PLP-RASTA and SDC7-1-3-7, 5s decision interval

Chapter 4

Conclusion

The tasks performed are not easy to solve completely and yet there exist no solution that does. A 100 percent solution will probably never be found, but one can try to get as near zero error rate as possible. The results of the music/speech/other solution gave very satisfactory results even for short decision intervals. Also sound samples taken from the Internet were tried with approximately the same result and because of the great results no other solutions were sought.

When it comes to the language identification problem the results were not so satisfactory. This task is probably even a harder task than music/speech/other discrimination but there exist solutions with far better results achieved here(with higher number of mixtures). The main problem here was that training with a high number of mixtures was approximately impossible due to memory issues and it is no doubt that higher number of mixtures were necessary to model the feature space in a decent way. This was partially showed introducing one language with 96 mixtures verses 64 for the others. It then became evident that the GMM was able to model the feature space for this 96 mixture language more accurately. Other than more mixtures, accuracy gain could have been achieve with a gender detector as a front end. Also a Universal Backend Model(UMB) probably would have gained the accuracy. It is a mixture model with typically over 1024 mixtures containing data for all languages. Data is then extracted, using a statistical method, from this all-in one model to form models for each language. A voiced/unvoiced decision module for speech in the front-end may also have increased the accuracy.

Chapter 5

Source Code

Here follows the implementations of the Matlab files used in the project. A more detailed description is found in the matlab files.The toolboxes mentioned in the text are required to run the files.

Function delta.m

```
function p=delta(signal)

%functions to make delta MFCC,PLP,LPC
%Deltac(k) = c(k+2)-c(k-2)
%Signal is MFCC coeff
%Returns the same number of deltas as number of MFCCs

%Author Tommy Strmhaug, 2008

%init return value
deltas=[];

%dimension of signal
[row,col]=size(signal);


for k=1:col

    %Make some trix at the beginning to be able to return same number of
    %deltas as number of input
    if k<3
        deltas=cat(2,deltas,signal(:,k+2)-signal(:,k+1));
    else

        %Make some trix at the beginning
        if k>col-2
            deltas=cat(2,deltas,signal(:,k)-signal(:,k-2));
        else
            deltas=cat(2,deltas,signal(:,k+2)-signal(:,k-2));

        end
    end

end

p=deltas;
```

Function deltaDelta.m

```
function p=deltadelta(signal)

%functions to make delta delta MFCC,PLP,LPC
%Deltadeltac(k) = delta c(k+1)-c(k-1)
%Signal is delta of MFCC
%Returns the same number of deltas as number of MFCCs

%Author Tommy Strmhaug, 2008

%init return value
deltas=[];

%dimension
[row,col]=size(signal);


for k=1:col

    %Make some trix at the beginning to be able to return same number of
    %deltas as number of input
    if k<2
        deltas=cat(2,deltas,signal(:,k+2)-signal(:,k+1));
    else

        %Make some trix at the beginning
        if k>col-1
            deltas=cat(2,deltas,signal(:,k)-signal(:,k-2));
        else

            deltas=cat(2,deltas,signal(:,k+1)-signal(:,k-1));

        end
    end

end

p=deltas;
```

Function feature_warp.m

```
function a = feature_warp(N,signal)

% Function to return warped features based on algorithm from the
% paper : Feature Warping for Robust Speaker Verification by
% Jason Pelecanos and Sridha Sridharan, 2001
%
% N is window width in number of features(MFCC, PLP) and
%could be seconds long(usually 3 sec).
% It also have to be an odd number for simplicity.
% signal is an array(not matrix) of MFCC,PLP features
% Return the new warped feature based on a gaussian distribution

% Author Tommy Strmhaug,2008
```

```
% New array will have same length as input due to padding

% Check if number is odd
if mod(N,2)==0
    error('N must be odd');
end

centerFeat = ceil(N/2);

%pad signal in a circular fashion
signal=signal;
padlength = centerFeat-1;
length(signal);
%pause(inf)

left=signal(1:padlength);
right=signal(end-padlength+1:end);
signal=cat(1,left,signal);
signal=cat(1,signal,right);

%Allocate new feature array for faster execution
%feats = zeros(1,length(signal));

for i=1:length(signal)-N+1

    %Find rank of center
    centerNumber = signal(i+centerFeat-1);
    % Current window
    %temparray=signal(i:(N-1)+i)
    %Sort array
    temparray=sort(signal(i:(N-1)+i),'descend');
    %Find rank of center window in the sorted list
    rank = find(temparray==centerNumber,1);

    feats(i)=warp(rank,N);

end
%dimesnion

a=feats;

x = -2.9:0.1:2.9;

%plot histogram
%hist(feats,40);

function b=warp(R,N)

%Function to make gaussian distributin

% Gaussian expression to warp to
%h = (1/sqrt(2*pi))*exp(-x^(2)/2);

% Solving the integral equation with respect to upper
% bound of integral. This is equation (5) in the paper.

x1 = norminv((N+0.5-R)/(N),0,1);
b=x1;
%b=solve((int(h,-inf,r)-((N+0.5-R)/(N))))
```

function make_gmm.m

```
function [mix, options, errlog] = make_gmm()

%Make a GMM to describe the distribution given by feature
%files on disc usually named 'speech.dat' and 'music.dat'
%or names of language files :
%'engn.dat' or 'germ.dat'.
%The function saves the model to disc.
%
%Author Tommy Strmhaug,2008

%Number of times to iterate
for it=1:1

    if(it==1)
        %Load feature sets
        [feat, t, nin, nout, ndata] = datread('fraq.dat');

        category ='fraq_silence_removed_7_PLP_rasta_SDC7_1_3_7';
    else
        if(it==2)
            [feat, t, nin, nout, ndata] = datread('germ.dat');
            category =
            'germ_silence_removed_MFCC_8_Feature_warping_SDC7_1_3_7';

        else
            [feat, t, nin, nout, ndata] =
            datread('othersounds-7-5-sec.dat');
            category ='othersoundsdiagS7mfccDC7-1-3-7';
        end

    end

    size(feat)
    tic
    %feat=single(feat);
    %Now make GMM approximations to the data distribution
    %using various numbers of mixtures.If planning to make
    %models with different number of mixtures
    %one can fill the number in in the array below.

    %Number of mixtures

    values=[64];

    for it=1:length(values)

        filenamemix = int2str(values(it))
        filename = [category filenamemix];

        ncentres = values(it)

        [inputdim,M] = size(feat);
        inputdim=M;

        %Number of EM-algorithm

        itr = 50;

        %NetLab GMM
```

```
        mix = gmm(inputdim, ncentres, 'diag');
        options = foptions;

        options(1) = 1; % Display error values.

        %minimum change in log-likelihood
        %options(3) = .1;

        % Maximum number of iterations for K-means algorithm.
        options(14) = 50;
        disp('Initializing GMM with k-means');
        mix = gmminit(mix, feat, options);
        % Display error values.
        options(1) = 1;
        % Prevent Covar values from collapsing
        %options(5) = 1;

        % Max. number of iterations
        options(14) = itr;
        disp('Running EM for mixture model');
        [mix, options, errlog] = gmmem(mix, feat, options);

        %Save model
        save (filename, 'mix');
        x = [5.4 3.6] ;
        %prob = gmmprob(mix, x)
        %mix.nin
        toc
    end
end
```

function SDC.m

```
\LARGE Function SDC.m

%functions to compute shifted delta cepstral coefficients
%from MFCC and PLP
%
%Signal is coeff of MFCC or PLP
%Returns coloums of SDCs. if k=7, there are 7 coloums for each MFCC

%Author Tommy Strmhaug, 2008


%d and P considered in frames. N amd k is ints
%N=7;  % number of c cepstral coefficients in each cepstral vector
%d=1;  % time advance and delay for the delta computation
%P=3;  % timeshift netween consecutive blocks
%k=7;  % number of blocks whose delta coefficients are
%concatenated to form the SDC vector

%dimension of signal
[row,col]=size(signal);

%init variables
sdc=[];
sdc_temp=[];

%Pad data to not get out of bound. circular padding
```

```
right=signal(:,col-9:col);
left =signal(:,1:21);
signal=cat(2,signal,left);
signal=cat(2,right,signal);

%time watch
tic

for t=11:col+10
    for i=0:k-1

        sdc_temp=signal(:,t+i*P+d)-signal(:,t+i*P-d);
        sdc=cat(2,sdc,sdc_temp);

    end

end
toc
p=sdc;
```

function normalize_vol.m

```
function res = normalize_vol(stream)

%function to normalize volume for speech samples

%Author Tommy Strmhaug, 2008

min_val=0;
max_val=max(stream)

%Preallocate
newStream=zeros(1,length(stream));

%volume factor
max_amplitude=0.5;

%Normalize
for x=1:length(stream)
    newStream(x)=((stream(x)-min_val)/(max_val-min_val))*max_amplitude;
end
res=newStream;
```

function removesilence.m

```
function E = remove_silence(signal, windowLength,step);

%Simple function for removing silence from sound files based
%on the RMS energy in the frame.
%The threshold is an experimental.

%Author Tommy Strmhaug, 2008

%RMS
signal = signal / max(max(signal));
```

```
signal2=signal;
curPos = 1;
L = length(signal)

numOfFrames = floor((L-windowLength)/step) + 1
%H = hamming(windowLength);
%E = zeros(numOfFrames,1);
for (i=2:numOfFrames)

    window = (signal(curPos:curPos+windowLength-1));
    E = (1/(windowLength)) * sum(abs(window.^2));
     if(abs(E)<0.00008)
         signal2(curPos:curPos+windowLength-1)=0;

     end
         curPos = curPos + step;
end

newsignal=signal2(signal2~=0);
E=newsignal;
%wavplay(newsignal(1:80000),8000)
```

function feat_extract.m

```
function []= feat_extract();

%Function to extract short time RMS and silence with application to
%speech/music discrimination.Function takes no paramters and dont
%return anything, but it writes feature vectors to disk.

%Author Tommy Strmhaug, 2008

%location of train files
dname=('speech');

%Iterate over both speech and music files

for i=1:1
    tempX=0;
    if (i~=1)
        dname='music';
    end

    %catalog containing training files
    dir2 = ['C:\matlab\musicspeech\wavfile\train\' dirname '\'];

    %Store files and catalogs in a temp variable
    a=dir(dir2);

    %No overlap
    overlap =0;

    %Open file for writing and add 'svm' for not to confuse
    %with feat-extract routines.

    fid = fopen([dirname 'svm' '.dat'],'w','a');
    %remover . and .. from file list
    for x=3:4

        filename=[dir2 a(x).name];
```

```
%read in file on by one
[signal,sr]=wavread(filename);

%samples/millisecond
sm = sr/1000;

%20 millisecond window in samples
windowlength = floor(sm*20);

%total number of frames
lengthFrames=floor(length(signal)/windowlength);

%feature length in millisecond. This is a desicion window
fl=2000;
%feature length in frames. This is a desicion window
featureLengthFrames=(sm*fl)/(sm*20);

if overlap
    delta = windowlength/5;
else
    delta = windowlength;
end

%Help variable
index = 0;

%make copy of original signal
s=signal;

%inital number of silence windows
silenceFrames=0;
nonSilenceFrames=0;

%RMS calculation
signal = signal.^2; % Square the samples
for i = 1:delta:length(signal)-windowlength+1
    index = index+1;
    % Average and take the square root of each window
    temp(i) = sqrt(mean(signal(i:i+windowlength-1)));
    if (abs(temp(i))<0.0010)
        y(index)=0;
        s(i:i+windowlength)=0;
    else
        y(index)=temp(i);
    end
end

%subplot(3,1,1);
%plot(s,'-r')

%hold on;
%subplot(3,1,2);
%plot(temp)

%number of desicion windows
numberOfD = floor(lengthFrames/featureLengthFrames);

%init
highEnergyFrames=0;
lowEnergyFrames=0;

%Put each feature frame into an array
```

```
        totalVar=[];

        %Find silence frames
        for x1=1:numberOfD

            frames=y(1:featureLengthFrames);
            variance=var(frames);

            totalVar(end+1)=variance;
            average=mean(frames);
            for x2=1:length(frames)

                if (frames(x2)/average)<0.15
                    lowEnergyFrames=lowEnergyFrames+1;
                else
                    highEnergyFrames=highEnergyFrames+1;
                end

                if frames(x2)==0
                    silenceFrames=silenceFrames+1;
                else
                    nonSilenceFrames=nonSilenceFrames+1;
                end

            end
            %fprintf(fid,'%f,%s\n',lowEnergyFrames/(lowEnergyFrames+
            highEnergyFrames),'speech');
            %Write to file for use with WEKA
            fprintf(fid,'%f,%f,%f,%s\n',variance,silenceFrames/
            (silenceFrames+nonSilenceFrames),lowEnergyFrames/
            (lowEnergyFrames+highEnergyFrames),dirname);
            silenceFrames=0;
            nonSilenceFrames=0;
            highEnergyFrames=0;
            lowEnergyFrames=0;
            y(1:featureLengthFrames)=[];
            frames=[];
            variance=0;
            average=0;

        end
        %subplot(3,1,3);
        %plot(totalVar,'-rs')

    end

    %close file
    fclose(fid);

end
```

function feat_extract2.m

```
function [frames,Z, s,x]= feat_extract2();

%function to extract features and save to files :
%speech.dat and music.dat othersounds2.dat
```

```
error=0;

%Parameter adjusting, 1 means true, 0 false

%number of Melfcc coeff to extract
numCoeff=8;
%use melfcc(1) or not(0)
mel=1;

%remove power term
remove_power=0;

%delta
addDelta=0;
deltas=[];

%deltadelt
addDeltaDelta=0;
deltadeltas=[]

%SDC
%Dont use delta or deltadelta or rmover power term with
%this option. However one can use mel + SDC. With this
%option N Melfcc are added to the SDC.
%Remove power is done automatic with this option

N=7;  % number of cepstral coefficients in each cepstral vector
d=1;  % time advance and delay for the delta computation
P=3;  % timeshift netween consecutive blocks
k=7;  % number of blocks whose delta coefficients are concatenated to
%form the SDC vector

SDC_coeff=1;
SDC_c=[];

%variable to hold total coeff
totalCoeff=0;

features=[];

%name of directory for train files. This also become the filename of the
%model
dname=('speech');

%Iterate over both speech and music files
for i=1:3
tempX=0;
if (i==2)
    dname='music';
end

if(i==3)
    dname='othersounds2';
end


%Path to trainfiles music and speech
if (i==1 || i==2)
dir2 = ['d:\matlab\musicspeech\wavfile\train\' dname '\']
else
    dir2=['d:\matlab\' dname '\BBC train\']
end
```

```
%read all names in dir
a=dir(dir2);

%Open file for writing
fid = fopen([dname '.dat'],'w','a');
%index start at 3 to remover . and .. from file list
for x=3:length(a);

    sprintf('%s %2.0f','Processing trainfile :',x-2)
    %filename of training file
    filename=[dir2 a(x).name];
    %read wave
    [signal,sr]=wavread(filename);
    %check size, files are 22kHz and use only first 4
    %seconds if files are bigger than 4 seconds
    length(signal);
    if (length(signal)>(88200))
        signal=signal(1:(88200));
    end
    length(signal);


    if(mel==1)
    coeff1=melfcc(signal,sr,'numcep', numCoeff);
    totalCoeff=numCoeff;
    end

   if(remove_power==1)
        %remove first coeffient
        coeff1(1,:)=[];
        totalCoeff=numCoeff-1;
   else
        totalCoeff=numCoeff;
   end



   if(SDC_coeff==1 && mel==1)
       coeff1=melfcc(signal,sr,'numcep', numCoeff);
       coeff1(1,:)=[];
       %make a copy of MFCCs
       coeff2=coeff1;
       totalCoeff=numCoeff-1;
       totalCoeff=(totalCoeff*k);
       SDC_c=SDC(coeff1,N,d,P,k);
       %to get correct length
       coeff1=SDC_c;

       %SDC_c(1:100)
   else

       if(SDC_coeff==1 && ~(mel==1))
       coeff1=melfcc(signal,sr,'numcep', numCoeff);
       coeff1(1,:)=[];

       totalCoeff=numCoeff-1;
       totalCoeff=totalCoeff*k;
       SDC_c=SDC(coeff1,N,d,P,k);
       %to get correct length
       coeff1=SDC_c;
       %SDC_c(1:100)
```

```
     end
   end


    if(addDelta)==1
        deltas = delta(coeff1);

    end

    if(addDeltaDelta)==1
    deltadeltas=deltadelta(deltas);

    end


    %samples/millisecond
    sm = sr/1000;
    %20 millisecond window in samples
    windowlength = floor(sm*20);

%total number of frames
lengthFrames=floor(length(signal)/windowlength);

%feature length in millisecond. This is a desicion window
fl=2000;
%feature length in frames. This is a desicion window
featureLengthFrames=(sm*fl)/(sm*20);

%Number of chunks of mfccs
%coeff1=coeff1(:);
[r,c]=size(coeff1);
r*c;
len=r*c;
totalCoeff;

%Check that MFCC length is legal, if not cut to right length.
if (mod(len,totalCoeff)~=0)
    error=error+1;

    re=floor(len/totalCoeff);
    len=re*totalCoeff;
end



coeff1;

%help variable
j=1;
error
%make feature file
for i2=1:totalCoeff:len
    %Write flag
    writeFlag=1;



   if(mel && ~SDC_coeff)
         %check and discard NaN feature lines
       if (any(isnan(coeff1(i2:i2+totalCoeff-1))) ||
       any(isnan(deltas(i2:i2+totalCoeff-1))) ||
       any(isnan(deltadeltas(i2:i2+totalCoeff-1))));
```

```
            %Do not write to file
            writeFlag=-1;
        end
        features =[coeff1(i2:i2+totalCoeff-1)];


         if(addDelta==1)
             features =[coeff1(i2:i2+totalCoeff-1)
             deltas(i2:i2+totalCoeff-1)];
         end

         if(addDeltaDelta==1)
                features =[coeff1(i2:i2+totalCoeff-1)
                deltas(i2:i2+totalCoeff-1)
                 deltadeltas(i2:i2+totalCoeff-1)];
          end
    end




     if(SDC_coeff==1 && mel==1)

        %check and discard NaN feature lines
        if (any(isnan(coeff2(j:j+(numCoeff-1)-1))) ||
        any(isnan(SDC_c(i2:i2+(totalCoeff)-1))))
            %Do not write to file
            writeFlag=-1;
        end

        features =[coeff2(j:j+(numCoeff-1)-1) SDC_c(i2:i2+
        (totalCoeff)-1) ];

     else
      if(SDC_coeff==1 && ~(mel==1))

          features =[SDC_c(i2:i2+(totalCoeff)-1)];
      end
     end

     j=j+7;



    if(writeFlag~=-1)
    fprintf(fid,'%f ',features,6);
    fprintf(fid,'\n');
    end
end


end


end

fclose(fid);
```

function feat_extract3.m

```
function []= feat_extract3();

%Function to extract features for language identification
%and save to file This file is kind of messy because there
%are little internal control, so it is possible to run
%illegal combinations of parameters. Each parameter
%has a comment of the most important restrictions.
%An example is that one should not run PLP and MFCC at t
%he same time. One also have to make sure that the parameters
%are equal for this function and the test function.

%Parameter adjusting are marked with boxes of %s, 1 means true, 0 false

%A variable to keep track of errors
error=0;

%delta parameter
di=0;

%%%%%%%%%%
%PLP-Rasta%
%%%%%%%%%%
%Do not use this together with MFCC. Do not use remove power
%either. When using PLP together with SDC ->order*k is the
%number of concatenated deltas to form the SDC.

PLP=1;
%Rasta filtering
rasta=1;
%model order. Remember that order+1 is number of coefficients
order=7;

%%%%%%
%MFCC%
%%%%%%
mel=0;
%Number of MFCC coeffs to extract. This is the number before removing the
%power term. Do not use together with PLP
numCoeff=8;

%%%%%%%%%%%%%%%%%%%
%Remove power term%
%%%%%%%%%%%%%%%%%%%
remove_power=0;

%%%%%%%%%%%%%%%%
%Remove Silence%
%%%%%%%%%%%%%%%%
remove_silence=1;

%%%%%%%%%%%%%%%%%
%Feature Warping%
%%%%%%%%%%%%%%%%%
featureWarp=0;
%time in seconds for feature warp window
time=3;
%Window length in number of features(default step time for MFCC is 10 ms)

if(mel==1)
    windowLength=numCoeff*time*100+1;
```

```
end

if(PLP==1)
    windowLength=(order+1)*time*100+1;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%
%Cepstral Mean Substraction%
%%%%%%%%%%%%%%%%%%%%%%%%%%
CMS=0;

%%%%%%%%%%%%%%%%%%
%Delta coefficients%
%%%%%%%%%%%%%%%%%%
addDelta=0;
deltas=[];

%%%%%%%%%%%%%%%%%%%%%%%%
%Delta delta coefficients%
%%%%%%%%%%%%%%%%%%%%%%%%
addDeltaDelta=0;
deltadeltas=[]

%%%%%
%SDC%
%%%%%

%Dont use delta or deltadelta or rmover power term with this option.
% However one can use mel + SDC. With this option N MFCC are added
%to the SDC. Remove power is done automatic with this option

N=7;  % number of cepstral coefficients in each cepstral vector
d=1;  % time advance and delay for the delta computation
P=3;  % timeshift netween consecutive blocks
k=7;  % number of blocks whose delta coefficients are
%concatenated to form the SDC vector

SDC_coeff=1;
SDC_c=[];


%variable to hold total coeff
totalCoeff=0;

%Vector of features
features=[];

%Iterate over language files
for i=1:5

    %Directory of training files. This is also the filename of the feature
    %files

    dname=('engn');

    tempX=0;

    if (i==2)
        dname='germ';
    end
    if(i==3)
        dname='japn';
```

```
end

if(i==4)
    dname='span'
end

if(i==5)
    dname='fraq'
end


% if(i==3)
%     dname='othersounds2';
% end


%Path to training files
dir2 = ['e:\CallFriend\train\' dname '\']

%Read all names in dir
a=dir(dir2);

%Open file for writing
fid = fopen([dname new '.dat'],'w','a');

%index start at 3 to remove . and .. from file list
for x=3:length(a);

    sprintf('%s %2.0f','Processing trainfile :',x-2)

    %filename of training files
    filename=[dir2 a(x).name];

    %read wave
    [signal,sr]=wavread(filename);

    %remove silence from first 5 minutes. will never use more
    if(remove_silence)
        'removing silence'
        signal=removesilence(signal(1:2400000), 20,10);
    end

    %All files are approx 30 minutes, some are shorter.
    %Extract 80 seconds data from each file.
    %SR = 8000 for speech files
    if (length(signal)>(480000))
        signal=signal(1:(480000));
    end

    if(PLP==1 && ~(SDC_coeff)==1 && ~(mel==1))
        'plp not SDC'
        % Calculate orderth order PLP features with/without RASTA
        [coeff1, spec2] = rastaplp(signal, sr, rasta, order);
        totalCoeff=order+1;
        if(remove_power==1)
            'remover power'
            %Remove first coefficient
            coeff1(1,:)=[];
            %Update total
            totalCoeff=totalCoeff-1;
        end
        if(featureWarp)
```

```
        'feature warp'
        coeff1=feature_warp(windowLength,coeff1(:));
    end


    coeff1;
end

if(PLP==1 && (SDC_coeff)==1 && ~(mel==1))
    'plp and SDC'
    % Calculate orderth order PLP features with/without RASTA
    [coeff1, spec2] = rastaplp(signal, sr, rasta, order);
    totalCoeff=order+1;
    coeff1(1,:)=[];
    %make a copy of MFCCs
    coeff2=coeff1;
    %Update
    totalCoeff=totalCoeff-1;
    di=totalCoeff
    numCoeff=totalCoeff;
    totalCoeff=(totalCoeff*k);
    SDC_c=SDC(coeff1,N,d,P,k);
    %to get correct length
    coeff1=SDC_c;

    if(featureWarp)
        'feature warp'
        coeff2=feature_warp(windowLength,coeff2(:));
    end


    coeff1;
end

if(mel==1 && ~(SDC_coeff)==1)
    'mel not SDC'
    %Extract numCoeff melfcc components with default settings
    coeff1=melfcc(signal,sr,'numcep', numCoeff);
    if(remove_power==1)
        'remover power'
        %Remove first coefficient
        coeff1(1,:)=[];
        %Update total
        totalCoeff=totalCoeff-1;


    end
   if(featureWarp)
        'feature warp'
        coeff1=feature_warp(windowLength,coeff1(:));
    end
end




if(SDC_coeff==1 && mel==1 && ~(PLP==1))
    'SDC and Mel'
    coeff1=melfcc(signal,sr,'numcep', numCoeff);
    coeff1(1,:)=[];
    %make a copy of MFCCs
    coeff2=coeff1;
```

```
        %Update
        totalCoeff=numCoeff-1;
        di=totalCoeff;
        totalCoeff=(totalCoeff*k);
        SDC_c=SDC(coeff1,N,d,P,k);
        %to get correct length
        coeff1=SDC_c;

        if(featureWarp)
            'feature warp'
            coeff2=feature_warp(windowLength,coeff2(:));
        end
        %SDC_c(1:100)
    end

    if(SDC_coeff==1 && ~(mel==1) && ~(PLP==1))
        'SDC and not mel'
        coeff1=melfcc(signal,sr,'numcep', numCoeff);
        coeff1(1,:)=[];
        totalCoeff=numCoeff-1;
        totalCoeff=totalCoeff*k;
        SDC_c=SDC(coeff1,N,d,P,k);
        %to get correct length
        coeff1=SDC_c;
        if(featureWarp)
            coeff1=feature_warp(windowLength,coeff1(:));
        end
        %SDC_c(1:100)
    end


if(addDelta)==1
    deltas = delta(coeff1);
    if(featureWarp)
        deltas=feature_warp(windowLength,deltas(:));
    end
end

if(addDeltaDelta)==1
    deltadeltas=deltadelta(deltas);
    if(featureWarp)
        deltadeltas=feature_warp(windowLength,deltadeltas(:));
    end

end

%Size of signal
[r,c]=size(coeff1);
r*c;
len=r*c;
totalCoeff;

%Check that MFCC length is legal, if not cut to right length.
if (mod(len,totalCoeff)~=0)
    error=error+1;
    re=floor(len/totalCoeff);
    len=re*totalCoeff;
end

%Helper variable
j=1;
```

```
%Make feature file
for i2=1:totalCoeff:len
    %Write flag
    writeFlag=1;

    if((PLP==1) && ~(SDC_coeff==1))
        %check and discard NaN feature lines
        if (any(isnan(coeff1(i2:i2+totalCoeff-1))) )
            writeFlag=-1;
        end
        features =[coeff1(i2:i2+totalCoeff-1)];
        %                if(i2>1)
        %                    pause(inf)
        %                end


    end
    if((PLP==1) && (SDC_coeff==1))
        %check and discard NaN feature lines
        if (any(isnan(coeff2(j:j+(numCoeff)-1))) ||
        any(isnan(SDC_c(i2:i2+(totalCoeff)-1))))
            %Do not write to file
            writeFlag=-1;
        end
        features =[coeff2(j:j+(numCoeff)-1) SDC_c(i2:i2+
        (totalCoeff)-1) ];



    end


    if(mel && ~SDC_coeff)
        %check and discard NaN feature lines
        if (any(isnan(coeff1(i2:i2+totalCoeff-1))) )
            writeFlag=-1;
        end

        features =[coeff1(i2:i2+totalCoeff-1)];

        if(addDelta==1)
            features =[coeff1(i2:i2+totalCoeff-1) deltas(i2:i2+
            totalCoeff-1)];
            if (any(isnan(deltas(i2:i2+totalCoeff-1))))
                writeFlag=-1;
            end
        end

        if(addDeltaDelta==1)
            features =[coeff1(i2:i2+totalCoeff-1) deltas(i2:i2+
            totalCoeff-1)
             deltadeltas(i2:i2+totalCoeff-1)];
            if (any(isnan(deltadeltas(i2:i2+totalCoeff-1))))
                writeFlag=-1;
            end
        end
    end
```

```
        if(SDC_coeff==1 && mel==1)

            %check and discard NaN feature lines
            if (any(isnan(coeff2(j:j+(numCoeff-1)-1))) ||
            any(isnan(SDC_c(i2:i2+(totalCoeff)-1))))
                %Do not write to file
                writeFlag=-1;
            end
            features =[coeff2(j:j+(numCoeff-1)-1) SDC_c(i2:i2+
            (totalCoeff)-1)];

        end


        if(SDC_coeff==1 && ~(mel==1) && ~(PLP==1))
                features =[SDC_c(i2:i2+(totalCoeff)-1)];
        end


        j=j+di;

        %Write features to file
        if(writeFlag~=-1)
            fprintf(fid,'%f ',features,6);
            fprintf(fid,'\n');
        end
    end


end


end

fclose(fid);
```

function test_gmm.m

```
function [mu sp]= test_gmm();

%function to test for speech/music and other

%Author Tommy Strmhaug, 2008

%Parameters are marked with boxes of %

features=[];

%decision length in duration*10 milliseconds
duration = 100

%variables related to error computation
totalError=0;
speechError=0;
musicError=0;
otherError=0;

total1=0;
total2=0;
```

```
total3=0;

errorInstancesM=0;
errorInstancesS=0;
errorInstancesO=0;

mu=[];
sp=[];

%iterate through all categories
for i=1:3

    %location of train files
    dname=('speech');
    if (i==2)
        dname='music\concatenated';
    end

    if(i==3)
        dname='D:\matlab\othersounds2\BBC test\'

    end

    globalSpeech=0;
    globalMusic=0;
    globalOther=0;


    %%%%%%%%%%%
    %use melfcc%
    %%%%%%%%%%%
    mel=1;
    %number of coeff in model.
    numCoeff=8;

    %%%%%%%%%%%%%%%%%%%
    %remove power term%
    %%%%%%%%%%%%%%%%%%%
    remove_power=1;

    %%%%%%%
    %delta%
    %%%%%%%
    addDelta=0;
    deltas=[];

    %%%%%%%%%%%
    %deltadelta%
    %%%%%%%%%%%
    addDeltaDelta=0;
    deltadeltas=[];

    %%%%%
    %SDC%
    %%%%%
    %Dont use delta or deltadelta or remover power term with
    %this option. However one can use mel + SDC. With this option
    %N Melfcc are added to the SDC. remove power is done automatic
    N=7;  % number of cepstral coefficients in each cepstral vector
    d=1;  % time advance and delay for the delta computation
    P=3;  % timeshift netween consecutive blocks
    k=7;  % number of blocks whose delta coefficients are
```

```
%concatenated to form the SDC vector
%use SDC
SDC_coeff=1;
SDC_c=[];

totalCoeff=0;

dir2 = ['d:\matlab\musicspeech\wavfile\test\' dname '\'];   %dname '\'
if(i==3)
    dir2=dname;
end

%read all names in dir
a=dir(dir2)

%load models
load speechdiag7mfccSDC7-1-3-764.mat
speech_model = mix;
mix
load musicdiag7mfccSDC7-1-3-764.mat
music_model = mix;
mix
load othersoundsdiag7mfccSDC7-1-3-7_7-5s_64-finished.mat
other_model=mix

speech=1;
music=1;
other=1;
s=0;
m=0;

%fid = fopen(['testspeech.dat'],'w','a');
%index start at 3 to remover . and .. from file list
for x=3:(length(a));
    sprintf('%s %2.0f :%s ','Processing testfile :',x-2,dname)

    filename=[dir2 a(x).name]
    %Load file
    [signal,sr]=wavread(filename);
    signal;
    %How much signa to use
    if (length(signal)>(100000))
        signal=signal(1:(100000));
    end

    if(mel==1)
        %Extract numCoeff melfcc components with default settings
        coeff1=melfcc(signal,sr,'numcep', numCoeff);
    end

    if(remove_power==1)
        %remove first coeffient
        coeff1(1,:)=[];
        totalCoeff=numCoeff-1;
    else
        totalCoeff=numCoeff;
    end

    if(addDelta)==1
        deltas = delta(coeff1);
    end
```

```
if(addDeltaDelta)==1
    deltadeltas=deltadelta(deltas);
end


if(SDC_coeff==1 && mel==1)
    coeff1=melfcc(signal,sr,'numcep', numCoeff);
    coeff1(1,:)=[];
    %make a copy of MFCCs
    coeff2=coeff1;
    totalCoeff=numCoeff-1;
    totalCoeff=(totalCoeff*k);
    SDC_c=SDC(coeff1,N,d,P,k);
    %to get correct length
    coeff1=SDC_c;
else

    if(SDC_coeff==1 && ~(mel==1))
        coeff1=melfcc(signal,sr,'numcep', numCoeff);
        coeff1(1,:)=[];
        totalCoeff=numCoeff-1;
        totalCoeff=totalCoeff*k;
        %SDC must have a vector
        SDC_c=SDC(coeff1,N,d,P,k);
        coeff1=SDC_c;
        %SDC_c(1:100)
    end
end


%Number of of mfccs
coeff1=coeff1(:);
coeff1=coeff1';

length(coeff1)/numCoeff;
range=1;
j=1;
for i2=1:totalCoeff:(length(coeff1))

    %     tempspeech = (gmmprob(speech_model,coeff1(i2:i2+12)));
    %     tempmusic = (gmmprob (music_model,coeff1(i2:i2+12)));
    if(mel==1)
        features =[coeff1(i2:i2+totalCoeff-1)];
    end

    if(addDelta==1)
        features =[coeff1(i2:i2+totalCoeff-1)
        deltas(i2:i2+totalCoeff-1)];
    end

    if(addDeltaDelta==1)
        features =[coeff1(i2:i2+totalCoeff-1)
        deltas(i2:i2+totalCoeff-1)
        deltadeltas(i2:i2+totalCoeff-1)];
    end

    if(SDC_coeff==1 && mel==1)


        features =[coeff2(j:j+(numCoeff-1)-1)
        SDC_c(i2:i2+(totalCoeff)-1) ];
```

```
else
    if(SDC_coeff==1 && ~(mel==1))

        features =[SDC_c(i2:i2+(totalCoeff)-1)];
    end
end

j=j+7;
%Probability density values
tempspeech = -log10(gmmprob(speech_model,features));

t=gmmprob(speech_model,features);

tempmusic = -log10(gmmprob (music_model,features));

t2=gmmprob (music_model,features);

tempother = -log10(gmmprob (other_model,features));

speech=speech+(tempspeech);

music=music+(tempmusic);

other=other+(tempother);

if range==duration

    %because a smal value have a higher log value less is
    %better

    if (speech<music)
        if(speech<other)
            globalSpeech=globalSpeech+1;
            'speech'
        else
            globalOther=globalOther+1;
            'other'
        end
    else
        if (music<other)
            globalMusic=globalMusic+1;
            'music'
        else
            globalOther=globalOther+1;
            'other'
        end
    end


    sp(end+1)=(speech);
    mu(end+1)=(music);
    speech;

    music;
    speech=1;
    music=1;
    other=1;

    range=0;
end

range=range+1;
```

```
        end

    end

    if(strcmp(dname, 'speech'))==1
        speechError = ((globalMusic+globalOther)/(globalSpeech+
        globalMusic+globalOther))*100
        errorInstancesM =errorInstancesM+globalMusic;
        errorInstancesO =errorInstancesO+globalOther;
        total1=globalMusic+globalSpeech+globalOther;
    end

    if(strcmp(dname, 'music\concatenated'))==1
        musicError = ((globalSpeech+globalOther)/(globalSpeech+
        globalMusic+globalOther))*100
        errorInstancesS =errorInstancesS+globalSpeech;
        errorInstancesO =errorInstancesO+globalOther;
        total2=globalMusic+globalSpeech+globalOther;
    end

    if(strcmp(dname, 'D:\matlab\othersounds2\BBC test\'))==1
        otherError = ((globalSpeech+globalMusic)/(globalSpeech+
        globalMusic+globalOther))*100
        errorInstancesS =errorInstancesS+globalSpeech;
        errorInstancesM =errorInstancesM+globalMusic;
        total3=globalOther+globalSpeech+globalMusic;
    end

end

globalSpeech
globalMusic

%Error report

speechError
musicError
otherError

totalError = (errorInstancesM+errorInstancesS+errorInstancesO)/
(total1+total2+total3)*100
```

function test_gmm2.m

```
function [mu sp]= test_gmm();

%Function to test language identification models.
%This file is kind of messy because there are little internal control, so
%it is possible to run illegal combinations of parameters. Each parameter
%has a comment of the most important restrictions. An example is that one
%should not run PLP and MFCC at the same time. One also have to make sure
%that the parameters are equal for this function and the test function.

%Parameter adjusting are marked with boxes of %s, 1 means true, 0 false
features=[];
%delta function

di=0;
%%%%%%%%%%%%%%%%
%Decision length%
```

```
%%%%%%%%%%%%%%%%
%decision length in duration*10 milliseconds
duration = 500

%language ident
%use engn,span,japn,germ,fraq for English,Spanish,
%Japanese, Deutsch and France
language1='engn';
language2='germ';
language3='span';
language4='japn';
language5='fraq';
%total number of languages
numberOfLanguages=5;

%variables related to error computation.
%Error1 = first language, Error2 = second ...
totalError=0;
Error1=0;
Error2=0;
Error3=0;
Error4=0;
Error5=0;

total1=0;
total2=0;
total3=0;
total4=0;
total5=0;

errorInstances2=0;
errorInstances1=0;
errorInstances3=0;
errorInstances4=0;
errorInstances5=0;

mu=[];
sp=[];

%confusion matrix variables
l11=0;
l12=0;
l13=0;
l14=0;
l15=0;

l21=0;
l22=0;
l23=0;
l24=0;
l25=0;

l31=0;
l32=0;
l33=0;
l34=0;
l35=0;

l41=0;
l42=0;
l43=0;
l44=0;
```

```
l45=0;

l51=0;
l52=0;
l53=0;
l54=0;
l55=0;

%Iterate through test dirs
for i=1:numberOfLanguages

    dname=language1;

    if(i==2)
        dname=language2;
    end

    if (i==3)
        dname=language3;
    end
    if (i==4)
        dname=language4;
    end
    if (i==5)
        dname=language5;
    end

    global1=0;
    global2=0;
    global3=0;
    global4=0;
    global5=0;

    %%%%%%%%%%
    %PLP-Rasta%
    %%%%%%%%%%
    %Do not use this together with MFCC. Do not use remove power
    %either. When using PLP together with SDC ->order*k is the number
    %of concatenated deltas to form the SDC.

    PLP=1;
    %Rasta filtering
    rasta=1;
    %model order. Remember that order+1 is number of coefficients
    order=7;

    %%%%%%
    %MFCC%
    %%%%%%
    mel=0;
    %Number of MFCC coeffs to extract. This is the number before
    %removing the power term. Do not use together with PLP
    numCoeff=8;

    %%%%%%%%%%%
    %Remove power term%
    %%%%%%%%%%%
    remove_power=0;

    %%%%%%%%%%%%%%%%%
    %Feature Warping%
    %%%%%%%%%%%%%%%%%
```

```
featureWarp=0;
%time in seconds for feature warp window
time=3;
%Window length in number of features
%(default step time for MFCC and PLP is 10 ms)
if(mel==1)
    windowLength=numCoeff*time*100+1;
end

if(PLP==1)
    windowLength=(order+1)*time*100+1;
end

%%%%%%%%%%%%%%%%%%%
%Delta Coefficients%
%%%%%%%%%%%%%%%%%%%
addDelta=0;
deltas=[];

%%%%%%%%%%%%%%%%%%%%%%%%%
%Delta delta Coefficients%
%%%%%%%%%%%%%%%%%%%%%%%%%
addDeltaDelta=0;
deltadeltas=[];

%%%%%%%%%%%%%%%
%Remove Silence%
%%%%%%%%%%%%%%%
remove_silence=1;

%%%%%
%SDC%
%%%%%
%Dont use delta or deltadelta or remover power term with this
%option. However one can use mel + SDC. With this option N Melfcc
%are added to the SDC. Remove power is done automatic
N=7;  % number of cepstral coefficients in each cepstral vector
d=1;  % time advance and delay for the delta computation
P=3;  % timeshift netween consecutive blocks
k=7;  % number of blocks whose delta coefficients are concatenated
%to form the SDC vector

SDC_coeff=1;
SDC_c=[];

%Total number of coefficients
totalCoeff=0;

%Dir for test files
dir2 = ['D:\CallFriend\test\' dname '\'];   %dname '\'

%Read all names in dir
a=dir(dir2);

%Load models
load engn_silence_removed_7_PLP_rasta_SDC7_1_3_7_64.mat
model1 = mix;

load germ_silence_removed_7_PLP_rasta_SDC7_1_3_7_64.mat
model2 = mix;

load span_silence_removed_7_PLP_rasta_SDC7_1_3_7_64.mat
```

```
model3 = mix;

load japn_silence_removed_7_PLP_rasta_SDC7_1_3_7_64.mat
model4 = mix;

load fraq_silence_removed_7_PLP_rasta_SDC7_1_3_7_64.mat
model5 = mix;

lang2=0;
lang1=0;
lang3=0;
lang4=0;
lang5=0;


%index start at 3 to remover . and .. from file list
for x=3:length(a);
    sprintf('%s %2.0f :%s ','Processing testfile :',x-2,dname)
    %Read filenames
    filename=[dir2 a(x).name];

    [signal,sr]=wavread(filename);

    %Remove silence from first 4 minutes. will never use more
    if(remove_silence)
        signal=removesilence(signal(1:1920000), 20,10);
        'removing silence'
    end

    %All files are approx 30 minutes, some are shorter.
    %Extract x seconds data from each test file.
    %SR = 8000 for speech files
    if (length(signal)>(280000))
        %make 4 desicions per test file, and dont start at the
        %beginning
        signal=signal(50000:(124000));
    end

    if(PLP==1 && ~(SDC_coeff)==1 && ~(mel==1))
        'plp not SDC'
        % Calculate orderth order PLP features with/without RASTA
        [coeff1, spec2] = rastaplp(signal, sr, rasta, order);
        totalCoeff=order+1;
        coeff1;

        if(remove_power==1)
            'remover power'
            %Remove first coefficient
            coeff1(1,:)=[];
            %Update total
            totalCoeff=totalCoeff-1;
        end
        if(featureWarp)
            'feature warp'
            coeff1=feature_warp(windowLength,coeff1(:));
        end
    end
    if(PLP==1 && (SDC_coeff)==1 && ~(mel==1))
        'plp and SDC'
        % Calculate orderth order PLP features with/without RASTA
        [coeff1, spec2] = rastaplp(signal, sr, rasta, order);
        totalCoeff=order+1;
```

```
    coeff1(1,:)=[];
    %make a copy of MFCCs
    coeff2=coeff1;
    %Update
    totalCoeff=totalCoeff-1;
    di=totalCoeff
    numCoeff=totalCoeff;
    totalCoeff=(totalCoeff*k);
    SDC_c=SDC(coeff1,N,d,P,k);
    %to get correct length
    coeff1=SDC_c;

    if(featureWarp)
        'feature warp'
        coeff2=feature_warp(windowLength,coeff2(:));
    end


    coeff1;
end


if(mel==1 && ~(SDC_coeff)==1)
    'mel not SDC'
    %Extract numCoeff melfcc components with default settings
    coeff1=melfcc(signal,sr,'numcep', numCoeff);
    if(remove_power==1)
        'remover power'
        %Remove first coefficient
        coeff1(1,:)=[];
        %Update total
        totalCoeff=totalCoeff-1;


    end
    if(featureWarp)
        'feature warp'
        coeff1=feature_warp(windowLength,coeff1(:));
    end
end


if(addDelta)==1
    deltas = delta(coeff1);
    if(featureWarp)
        deltas=feature_warp(windowLength,deltas(:));
    end
end

if(addDeltaDelta)==1
    deltadeltas=deltadelta(deltas);
    if(featureWarp)
        deltadeltas=feature_warp(windowLength,deltadeltas(:));
    end
end

if(SDC_coeff==1 && mel==1 && ~(PLP==1))
    'SDC and Mel'
    coeff1=melfcc(signal,sr,'numcep', numCoeff);
    coeff1(1,:)=[];
    %make a copy of MFCCs
    coeff2=coeff1;
```

```
    %Update
    totalCoeff=numCoeff-1;
    di=totalCoeff;
    totalCoeff=(totalCoeff*k);
    SDC_c=SDC(coeff1,N,d,P,k);
    %to get correct length
    coeff1=SDC_c;

    if(featureWarp)

        coeff2=feature_warp(windowLength,coeff2(:));
    end
    %SDC_c(1:100)
end

if(SDC_coeff==1 && ~(mel==1) && ~(PLP==1))
    'SDC and not mel'
    coeff1=melfcc(signal,sr,'numcep', numCoeff);
    coeff1(1,:)=[];
    totalCoeff=numCoeff-1;
    totalCoeff=totalCoeff*k;
    SDC_c=SDC(coeff1,N,d,P,k);
    %to get correct length
    coeff1=SDC_c;
    if(featureWarp)
        coeff1=feature_warp(windowLength,coeff1(:));
    end
    %SDC_c(1:100)
end


coeff1=coeff1(:);
coeff1=coeff1';

length(coeff1)/numCoeff;
'length'
length(coeff1)
totalCoeff
range=1;
j=1;

for i2=1:totalCoeff:(length(coeff1))

    %     tempspeech =
    %(gmmprob(speech_model,coeff1(i2:i2+12)));
    %     tempmusic =
    %(gmmprob (music_model,coeff1(i2:i2+12)));

    if(PLP==1 && ~(SDC_coeff==1))
        features =[coeff1(i2:i2+totalCoeff-1)];
        %                 if(i2>1)
        %                     pause(inf)
        %                 end

    end

    if((PLP==1) && (SDC_coeff==1))
        %check and discard NaN feature lines


        features =[coeff2(j:j+(numCoeff)-1)
        SDC_c(i2:i2+(totalCoeff)-1) ];
```

```
end


if(mel && ~SDC_coeff)
    %check and discard NaN feature lines

    features =[coeff1(i2:i2+totalCoeff-1)];

    if(addDelta==1)
        features =[coeff1(i2:i2+totalCoeff-1)
        deltas(i2:i2+totalCoeff-1)];

    end

    if(addDeltaDelta==1)
        features =[coeff1(i2:i2+totalCoeff-1)
        deltas(i2:i2+totalCoeff-1)
        deltadeltas(i2:i2+totalCoeff-1)];

    end
end

if(SDC_coeff==1 && mel==1)


    features =[coeff2(j:j+(numCoeff-1)-1)
    SDC_c(i2:i2+(totalCoeff)-1) ];

end


if(SDC_coeff==1 && ~(mel==1) && ~(PLP==1))
    features =[SDC_c(i2:i2+(totalCoeff)-1)];
end


j=j+di;

%Probability dinsities

temp1 = -log10(gmmprob(model1,features));

%t=gmmprob(engn_model,features);

temp2 = -log10(gmmprob (model2,features));

%t2=gmmprob (germ_model,features);

temp3 = -log10(gmmprob(model3,features));

temp4 = -log10(gmmprob(model4,features));

temp5 = -log10(gmmprob(model5,features));

lang1=lang1+(temp1);

lang2=lang2+(temp2);

lang3=lang3+temp3;

lang4=lang4+(temp4);
```

```
lang5=lang5+temp5;

if range==duration

    %because a smaller value have a higher log value
    %
    %                 if (lang1<lang2)
    %                     if(lang1<lang3)
    %
    %                     else
    %
    %
    %                         global3=global3+1;
    %                         language3
    %                     end
    %
    %
    %                 else
    %                     if (lang2<lang3)
    %                         global2=global2+1;
    %                         language2
    %                     else
    %                         global3=global3+1;
    %                         language3
    %                     end
    %                 end

    candidate = min(lang1,lang2);
    candidate2 =min(lang3,lang4);
    candidate3 =min(candidate,candidate2);
    candidate4= min(candidate3,lang5);

    if candidate4 == lang1
        global1=global1+1;
        language1
    end

    if candidate4 == lang2
        global2=global2+1;
        language2
    end

    if candidate4 == lang3
        global3=global3+1;
        language3
    end

    if candidate4 == lang4
        global4=global4+1;
        language4
    end

    if candidate4 == lang5
        global5=global5+1;
        language5
    end

    lang1=0;
    lang2=0;
    lang3=0;
    lang4=0;
```

```
            lang5=0;


            range=0;
        end

        range=range+1;

    end

end

if(strcmp(dname, language1))==1
    Error1 = ((global2+global3+global4+global5)/
    (global1+global2+global3+global4+global5))*100
    errorInstances2 =errorInstances2+global2;
    errorInstances3 =errorInstances3+global3;
    errorInstances4 =errorInstances4+global4;
    errorInstances5 =errorInstances5+global5;
    total1=global2+global1+global3+global4+global5;
    l11=total1-(global2+global3+global4+global5);
    l12=global2;
    l13=global3;
    l14=global4;
    l15=global5;

end
if(strcmp(dname, language2))==1
    Error2 = ((global1+global3+global4+global5)/
    (global1+global2+global3+global4+global5))*100
    errorInstances1=errorInstances1+global1;
    errorInstances3=errorInstances3+global3;
    errorInstances4=errorInstances4+global4;
    errorInstances5=errorInstances5+global5;

    total2=global2+global1+global3+global4+global5;

    l22=total2-(global1+global3+global4+global5);
    l21=global1;
    l23=global3;
    l34=global4;
    l35=global5;
end

if(strcmp(dname, language3))==1
    Error3 = ((global1+global2+global4+global5)/
    (global1+global2+global3+global4+global5))*100
    errorInstances1=errorInstances1+global1;
    errorInstances2=errorInstances2+global2;
    errorInstances4=errorInstances4+global4;
    errorInstances5=errorInstances5+global5;
    total3=global2+global1+global3+global4+global5;
    l33=total3-(global2+global1+global4+global5);
    l32=global2;
    l31=global1;
    l34=global4;
    l35=global5;

end

if(strcmp(dname, language4))==1
    Error4 = ((global1+global2+global3+global5)/
```

```
        (global1+global2+global3+global4+global5))*100
        errorInstances1=errorInstances1+global1;
        errorInstances2=errorInstances2+global2;
        errorInstances3=errorInstances3+global3;
        errorInstances5=errorInstances5+global5;
        total4=global2+global1+global3+global4+global5;
        l44=total4-(global2+global1+global3+global5);
        l41=global1;
        l42=global2;
        l43=global3;
        l45=global5;

    end

    if(strcmp(dname, language5))==1
        Error5 = ((global1+global2+global3+global4)/
        (global1+global2+global3+global4+global5))*100
        errorInstances1=errorInstances1+global1;
        errorInstances2=errorInstances2+global2;
        errorInstances3=errorInstances3+global3;
        errorInstances4=errorInstances4+global4;
        total5=global2+global1+global3+global4+global5;
        l55=total5-(global2+global1+global3+global4);
        l51=global1;
        l52=global2;
        l53=global3;
        l54=global4;
    end

    global1
    global2
    global3

end

%Error report
sprintf('%s %s :%f',language1,'error :',Error1)
sprintf('%s %s :%f',language2,'error :',Error2)
sprintf('%s %s :%f',language3,'error :',Error3)
sprintf('%s %s :%f',language4,'error :',Error4)
sprintf('%s %s :%f',language5,'error :',Error5)

totalError = (errorInstances2+errorInstances1+errorInstances3+
errorInstances4+errorInstances5)/
(total1+total2+total3+total4+total5)*100
%confusion
sprintf('%f %f %f %f %f',l11,l12,l13,l14,l15)
sprintf('%f %f %f %f %f',l21,l22,l23,l24,l25)
sprintf('%f %f %f %f %f',l31,l32,l33,l34,l35)
sprintf('%f %f %f %f %f',l41,l42,l43,l44,l45)
sprintf('%f %f %f %f %f',l51,l52,l53,l54,l55)
```

function cdp.m

```
function mini=cdp(ref, stream);

% Continuous DP algorithm for spotting reference frames
```

```
% in a stream.See article :
% Spotting Method for Classification of Real World Data, by Ryuichi Oka

% Author Tommy Strmhaug, 2008

clear P;

%init variables
tau=1;
mini=inf;
P=zeros(numel(ref),4);

%According to algorith these must be inf.
P(:,1)=inf;
P(:,2)=inf;

tic
for t2=3:numel(stream),
    t=t2;
    if t2>4
       t=4;
       P(:,1)=[];
    end

    while(tau<numel(ref)+1)
        if (tau==1)
            P(1,t)=3*abs(stream(t2)-ref(1));
        end

        if (tau==2)

            temp=min(P(1,t-2)+2*abs(stream(t2-1)-ref(2))+abs(stream(t2)-
            ref(2)),P(1,t-1)+3*abs(stream(t2)-ref(2)));
            P(2,t)= min(temp,P(1,t)+3*abs(stream(t2)-ref(2)));

        end

        if(tau>2)

            temp=min(P(tau-1,t-2)+2*abs(stream(t2-1)-
            ref(tau))+abs(stream(t2)-ref(tau)),P(tau-1,t-1)+
            3*abs(stream(t2)-ref(tau)));
            P(tau,t)= min(temp,P(tau-2,t-1)+3*abs(stream(t2)
            -ref(tau-1))+3*abs(stream(t2)-ref(tau)));

        end
    tau=tau+1;
    end

    if(P(tau-1,t)/(3*(tau-1))<mini)
        mini=P(tau-1,t)/(3*(tau-1));
    end

    p(t2)=P(tau-1,t)/(3*(tau-1));

    tau=1;

end

%At last plot the contour of spotting
plot(p)
```

```
toc
```

function extract_words.m

```
function []=extract_words()

%Function to split a file into words according to the trascription file

%Read all transcription files and wavefiles from each dialect directory and
%split files into word accordingly to trascription files
%Change variables manually for other files, like name, and number of lines
%in the .WRD file

%Author Tommy Strmhaug,2008

%Change this for other catalogs(DR1-DR8)
dialect='DR8\'

%This has to be changed for each dialect
dir2 = ['E:\speech samples\TIMIT\TRAIN\' dialect];

%read all names in dir
a=dir(dir2)

%Iterate all catalogs in dir2 path
for i=3:length(a)
ext=a(i).name

%Define paths
path1=['E:\speech samples\TIMIT\TRAIN\' dialect ext '\' 'SA1.wav']
path2=['E:\speech samples\TIMIT\TRAIN\' dialect ext '\' 'SA2.wav']
path3=['E:\speech samples\TIMIT\TRAIN\' dialect ext '\' 'SA1.WRD']
path4=['E:\speech samples\TIMIT\TRAIN\' dialect ext '\' 'SA2.WRD']

[wfile1,sr] = wavread(path1);
[wfile2,sr] = wavread(path2);

%obtain beginning and end of each word
[s1, e1, wrd1] = textread(path3,'%f %f %s', 11);
[s2, e2, wrd2] = textread(path4,'%f %f %s', 10);

%split file into words and save with same filename as word
for it=1:length(s1)

    b=['E:\speech samples\TIMIT\TRAIN\' dialect ext '\' 'SA1\' ]
    mkdir (b)
    name = strcat(b,wrd1{it},'.wav');

    temp=wfile1;
    newWord=temp(s1(it):e1(it));
    wavwrite(newWord,sr,name)
end

for it=1:length(s2)
    b=['E:\speech samples\TIMIT\TRAIN\' dialect ext '\' 'SA2\']
    mkdir (b)
    name = strcat(b,wrd2{it},'.wav');
```

```
    temp=wfile1;
    newWord=temp(s2(it):e2(it));
    wavwrite(newWord,sr,name)
end

end
```

function spectral_centroid.m

```
function c = spectral_centroid(signal,windowLength, step, fs)

%Function to calculate the spectral centroid
%Author Tommy Strmhaug, 2008

signal = signal / max(abs(signal));
l = length(signal);
numOfFrames = floor((l-windowlength)/step) + 1;
H = hamming(windowlength);
currentPos = 1;
m = ((fs/(2*windowlength))*[1:windowlength])';
c = zeros(numOfFrames,1);
for (i=1:numOfFrames)
    window = H.*(signal(currentPos:currentPos+windowlength-1));
    FFT = (abs(fft(window,2*windowlength)));
    FFT = FFT(1:windowlength);
    FFT = FFT / max(FFT);
    c(i) = sum(m.*FFT)/sum(FFT);
    if (sum(window.^2)<0.010)
        c(i) = 0.0;
    end
    currentPos = currentPos + step;
end
c = c / (fs/2);
```

function spectral_flux.m

```
function F = spectral_flux(signal,windowLength, step, fs)

%Function to calculate the spectral flux
%Author Tommy Strmhaug, 2008

signal = signal / max(abs(signal));
currentPos = 1;
l = length(signal);
numOfFrames = floor((l-windowlength)/step) + 1;
H = hamming(windowlength);
m = [0:windowlength-1]';
F = zeros(numOfFrames,1);
for (i=1:numOfFrames)
    window = H.*(signal(currentPos:currentPos+windowlength-1));
    FFT = (abs(fft(window,2*windowlength)));
    FFT = FFT(1:windowlength);
    FFT = FFT / max(FFT);
    if (i>1)
        F(i) = sum((FFT-FFTprev).^2);
    else
        F(i) = 0;
    end
    currentPos = currentPos + step;
```

```
    FFTprev = FFT;
end
```

function spectral_rollOff.m

```
function sR = spectral_rollOff(signal,windowlength, step, c, fs)

%function to calculate spectral rolloff
%the parameter c is given in percent.Spectral Rolloff is the frequency
%below which c=85 percent of spectrum distribution concentrated.

%Author Tommy Strmhaug, 2008

signal = signal / max(abs(signal));
currentPos = 1;
l = length(signal);
numOfFrames = (l-windowlength)/step + 1;
H = hamming(windowlength);
m = [0:windowlength-1]';
for (i=1:numOfFrames)
    window = (signal(currentPos:currentPos+windowlength-1));
    FFT = (abs(fft(window,512)));
    FFT = FFT(1:255);
    totalEnergy = sum(FFT);
    curEnergy = 0.0;
    countFFT = 1;
    while ((curEnergy<=c*totalEnergy) && (countFFT<=255))
        curEnergy = curEnergy + FFT(countFFT);
        countFFT = countFFT + 1;
    end
    sR(i) = ((countFFT-1))/(fs/2);
    currentPos = currentPos + step;
end
```

function spectral_rollOff.m

```
function E = short_time_energy(signal, windowLength,step);

%function to calculate short time energy

%Author Tommy Strmhaug, 2008

signal = signal / max(max(signal));
currentPos = 1;
l = length(signal);
numOfFrames = floor((l-windowlength)/step) + 1;
E = zeros(numOfFrames,1);
for (i=1:numOfFrames)
    window = (signal(currentPos:currentPos+windowlength-1));
    E(i) = (1/(windowlength)) * sum(abs(window.^2));
    currentPos = currentPos + step;
end
```

Bibliography

[1] *NETLAB: algorithms for pattern recognition.* Springer-Verlag New York, Inc., New York, NY, USA, 2002.
Software is found athttp://www.ncrg.aston.ac.uk/netlab/.

[2] Kiyoharu. Aizawa, Yuichi. Nakamura, and Shin'ichi Satoh. Satoh. Hmm-based audio keyword generation. *Advances in Multimedia Information Processing - PCM 2004: 5th Pacific Rim Conference on Multimedia*, 2004.

[3] F. Allen, E. Ambikairajah, and J. Epps. Warped magnitude and phase-based features for language identification. *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, 1:I–I, May 2006.

[4] Felicity Allen, Eliathamby Ambikairajah, and Julien Epps. Language identification using warping and the shifted delta cepstrum. *Multimedia Signal Processing, 2005 IEEE 7th Workshop on*, pages 1–4, Oct. 2005.

[5] S. Bhattacharyya, T. Srikanthan, and P. Krishnamurthy. Ideal gmm parameters and posterior log likelihood for speaker verification. *Neural Networks for Signal Processing XI, 2001. Proceedings of the 2001 IEEE Signal Processing Society Workshop*, pages 471–480, 2001.

[6] Paul. Boersma and David. Weenink. Praat: doing phonetics by computer (version 5.0.26).
http://www.praat.org/.

[7] S. Boll. Suppression of acoustic noise in speech using spectral subtraction. *Acoustics, Speech, and Signal Processing [see also IEEE Transactions on Signal Processing], IEEE Transactions on*, 27(2):113–120, Apr 1979.

[8] L. Burget, P. Matejka, P. Schwarz, O. Glembek, and J. Cernocky. Analysis of feature extraction and channel compensation in a gmm speaker recognition system. *Audio, Speech, and Language Processing, IEEE Transactions on*, 15(7):1979–1986, Sept. 2007.

[9] W.M. Campbell, J.P. Campbell, D.A. Reynolds, E. Singer, and P.A. Torres-Carrasquillo. Support vector machines for speaker and language recognition. *Computer Speech and Language*, 20(2–3):210–229, April–July 2006.

[10] W.M. Campbell, D.A. Reynolds, E. Singer, and P.A. Torres-Carrasquillo. Language recognition with support vector machines. *In Proc. Odyssey: The Speaker and Language Recognition Workshop in Toledo, Spain, ISCA*, pages 41–44, May 2004.

[11] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[12] Hoon-Young Cho, Sang-Mun Chi, and Yung-Hwan Oh. A robust front-end for telephone speech recognition. In *PRICAI '98: Proceedings of the 5th Pacific Rim International Conference on Artificial Intelligence*, pages 636–644, London, UK, 1998. Springer-Verlag.

[13] Wu Chou and Bing Huang Juang, editors. *Pattern Recognition in Speech and Language Processing.* CRC Press, Inc., Boca Raton, FL, USA, 2002.

[14] R. Christiansen and C. Rushforth. Detecting and locating key words in continuous speech using linear predictive coding. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 25(5):361–367, Oct 1977.

[15] Linguistic Data Consortium. Callfriend corpus, 1996.

[16] J. de Veth and L. Boves. Comparison of channel normalisation techniques for automatic speech recognition over the phone. *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*, 4:2332–2335 vol.4, Oct 1996.

[17] K. El-Maleh, M. Klein, G. Petrucci, and P. Kabal. Speech/music discrimination for multimedia applications. *Acoustics, Speech, and Signal Processing, 2000. ICASSP '00. Proceedings. 2000 IEEE International Conference on*, 6:2445–2448 vol.4, 2000.

[18] Yasser EL-Manzalawy and Vasant Honavar. *WLSVM: Integrating LibSVM into Weka Environment*, 2005. Software available at http://www.cs.iastate.edu/~yasser/wlsvm.

[19] Daniel P. W. Ellis. PLP and RASTA (and MFCC, and inversion) in Matlab, 2005.
online web resource
http://www.ee.columbia.edu/~dpwe/resources/matlab/rastamat/.

[20] Michael W. Eysenck and Mark T. Keane. *Cognitive Psychology: A Student's Handbook*. Psychology Press (UK), August 2000.

[21] van der Heijden. Ferdi, Robert. P.W. Duin, Dick. de Ridder, and David. M.J. Tax, editors. *Classification, parameter estimation and state estimation - an engineering approach using Matlab*. John Wiley and Sons, 2004.

[22] Lori F. Garofolo, John S.and Lamel, William M. Fisher, Jonathan G. Fiscus, David S. Pallett, Nancy L. Dahlgren, and Victor. Zue. Timit acoustic-phonetic continuous speech corpus, 1993. http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC93S1.

[23] Steven Greenberg. *Speech Processing in the Auditory System*. Springer-Verlag, New York, 2004.

[24] H. Greenspan and A.T. Pinhas. Medical image categorization and retrieval for pacs using the gmm-kl framework. *Information Technology in Biomedicine, IEEE Transactions on*, 11(2):190–202, March 2007.

[25] H. Hermansky, B. Hanson, and H. Wakita. Perceptually based linear predictive analysis of speech. *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '85.*, 10:509–512, Apr 1985.

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

[27] H. Hermansky, N. Morgan, Aruna Bayya, and Phil Kohn. RASTA-PLP speech analysis. Technical Report TR-91-069, 4001 Discovery Drive, Boulder, CO 80303, 1991. citeseer.ist.psu.edu/144042.html.

[28] Hynek Hermansky. Modulation spectrum in speech processing.

[29] Q.Y. Hong. and S. Kwong. A genetic classification method for speaker recognition. *Engineering Applications of Artificial Intelligence*, 18(1):13–19, February 2005.

[30] Y. Itoh, J. Kiyama, H. Kojima, S. Seki, and R. Oka. A proposal for a new algorithm of reference interval-free continuous dp for real-time speech or text retrieval. *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*, 1:486–489 vol.1, Oct 1996.

[31] Yoshiaki Itoh. Shift continuous dp: A fast matching algorithm between arbitrary parts of two time-sequence data sets. *Syst. Comput. Japan*, 36(10):43–53, 2005.

[32] Jyh-Shing Roger Jang. Audio processing toolbox.
available from the link at the author's homepage at :
http://www.cs.nthu.edu.tw/~jang.

[33] Garcia. Jayme, Barbedo. Arnal, and Lopes. Amauri. New feature for automatic speech/music discrimination. 2005.
http://gsd.ime.usp.br/sbcm/2005/papers/short-12351.pdf.

[34] Zoher Z. Karu. *Signals and Systems Made Ridiculously Simple.* Zi Zi Press, Cambridge, 2001.

[35] C. Kermorvant. A comparison of noise reduction techniques for robust speech recognition. IDIAP-RR 10, IDIAP, 1999. IDIAP-RR 99-10.

[36] M.A. Kohler and M. Kennedy. Language identification using shifted delta cepstra. *Circuits and Systems, 2002. MWSCAS-2002. The 2002 45th Midwest Symposium on*, 3:III–69–72 vol.3, Aug. 2002.

[37] Pascal Clark Les Atlas, Steve Schimmel. *University of Washington ISDL modulation toolbox*, 2008.
http://isdl.ee.washington.edu/projects/modulationtoolbox/.

[38] Haizhou Li and Bin Ma. A phonotactic language model for spoken language identification. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 515–522, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

[39] Peng Lim, San Quek, and Khiang Peh. Application of the gaussian mixture model to drug dissolution profiles prediction. *Neural Comput. Appl.*, 14(4):345–352, 2005.

[40] Xiabi Liu, Hui Fu, and Yunde Jia. Gaussian mixture modeling and learning of neighboring characters for multilingual text extraction in images. *Pattern Recogn.*, 41(2):484–493, 2008.
http://dx.doi.org/10.1016/j.patcog.2007.06.004.

[41] Modulations Nima Mesgarani. Speech discrimination based on multi-scale spectro–temporal.

[42] Tom M. Mitchell. *Machine Learning.* McGraw-Hill, New York, 1997.

[43] Chiyomi Miyajima, Yosuke Hattori, Keiichi Tokuda, Takashi Masuko, Takao Kobayashi, and Tadashi Kitamura. Speaker identification using gaussian mixture models based on multi-space probability distribution. *Proc. of ICASSP*, 84:847–855, 2001.

[44] P. Moerland. A comparison of mixture models for density estimation, 1999.
citeseer.ist.psu.edu/moerland99comparison.html.

[45] O. Mubarak, E. Ambikairajah, and J. Epps. Analysis of an analysis of an mfcc based audio indexing system for efficient coding of multimedia. *in Proc. The Eighth International Symposium on Signal Processing and its applications*, 2005.

[46] C. Myers, L. Rabiner, and A. Rosenberg. An investigation of the use of dynamic time warping for word spotting and connected speech recognition. *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '80.*, 5:173–177, Apr 1980.

[47] Cory R. Myers. A comparative study of several dynamic time warping algorithms for speech, 1980.

[48] Malayath. Narendranath, Hermansky. Hynek, Kajarekara. Sachin, and B. Yegnanarayana. Data-driven temporal filters and alternatives to gmm in speaker verification. *Digital Signal Processing, IEEE International Conference on ICASSP '85.*, 10(1):55–74, Jan 2000.

[49] Ryuichi Oka. Spotting method for classification of real world data. *Comput. J.*, 41(8):559–565, 1998.

[50] Antonio Peinado. *Speech Recognition over Digital Channels: Robustness And Standards*. John Wiley & Sons, 2006.

[51] Jason. Pelecanos and Sridha. Sridharan. Feature warping for robust speaker verification. *Proceedings 2001: A Speaker Odyssey - The Speaker Recognition Workshop*, 2001.
http://eprints.qut.edu.au/archive/00010408/.

[52] J.W. Picone. Signal modeling techniques in speech recognition. *Proceedings of the IEEE*, 81(9):1215–1247, Sep 1993.

[53] N.J Princeton. Bbc sound effects library (royalty free) - a comprehensive collection of sound effects, 1983, 91-97.
http://www.sound-ideas.com/bbc.html.

[54] L. Rabiner and S. Levinson. A speaker-independent, syntax-directed, connected word recognition system based on hidden markov models and level building. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 33(3):561–573, Jun 1985.

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

[56] J. Saunders. Real-time discrimination of broadcast speech/music. In *ICASSP '96: Proceedings of the Acoustics, Speech, and Signal Processing, 1996. on Conference Proceedings., 1996 IEEE International Conference*, pages 993–996, Washington, DC, USA, 1996. IEEE Computer Society.

[57] B. Sch, o Platt, J. Shawe-Taylor, A. Smola, and R. Williamson. Estimating the support of a high-dimensional distribution, 1999. citeseer.ist.psu.edu/sch01estimating.html.

[58] E. Scheirer and M. Slaney. Construction and evaluation of a robust multifeature speech/music discriminator. In *ICASSP '97: Proceedings of the 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '97)-Volume 2*, page 1331, Washington, DC, USA, 1997. IEEE Computer Society.

[59] Eric. Scheirer and Malcolm . Slaney. The music-speech corpus, 1996. http://labrosa.ee.columbia.edu/sounds/musp/scheislan.html.

[60] S. Schimmel and L. Atlas. Coherent envelope detection for modulation filtering of speech. *Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on*, 1:221–224, 18-23, 2005.

[61] Tanja Schultz and Katrin Kirchhoff. *Multilingual Speech Processing.* Academic Press, Inc., Orlando, FL, USA, 2006.

[62] Uma Srinivasan and Surya Nepal. *Managing Multimedia Semantics.* IRM Press, 2005.

[63] Michael Talbot-Smith, editor. *Audio Engineer's Reference Book.* Focal Press, 1999.

[64] D.M.J. Tax. Ddtools, the data description toolbox for matlab, Jun 2008. version 1.6.3.

[65] Inc The MathWorks. 1106 - memory management guide. http://www.mathworks.com/support/tech-notes/1100/1106.html.

[66] P.A. Torres-Carrasquillo, D.A. Reynolds, and Jr. Deller, J.R. Language identification using gaussian mixture model tokenization. *Acoustics, Speech, and Signal Processing, 2002. Proceedings. (ICASSP '02). IEEE International Conference on*, 1:I–757–I–760 vol.1, 2002.

[67] Vladimir N. Vapnik. *The nature of statistical learning theory.* Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[68] Janez Žibert, Nikola Pavesić, and France Mihelič. Speech/non-speech segmentation based on phoneme recognition features. *EURASIP J. Appl. Signal Process.*, 2006(1):47–47, uary.

[69] Alex Waibel and Kai-Fu Lee, editors. *Readings in speech recognition.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[70] M. Weintraub. Keyword-spotting using sri's decipher large-vocabulary speech-recognition system. *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*, 2:463–466 vol.2, Apr 1993.

[71] G. Williams and D. Ellis. Speech/music discrimination based on posterior probability features. In *Proc. Eurospeech*, Budapest, 1999.

[72] Ian H. Witten. and Eibe Frank., editors. *Data Mining: Practical machine learning tools and techniques, 2nd Edition.* Focal PresMorgan Kaufmanns, 2005.

[73] Bo Yin, Eliathamby Ambikairajah, and Fang Chen. Combining cepstral and prosodic features in language identification. In *ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition*, pages 254–257, Washington, DC, USA, 2006. IEEE Computer Society.

[74] K. Yiu, M. Mak, and C. Li. Gaussian mixture models and probabilistic decision-based neural networks for pattern classification: A comparative study, 1999.
citeseer.ist.psu.edu/yiu99gaussian.html.

[75] M.A. Zissman. Comparison of four approaches to automatic language identification of telephone speech. *Speech and Audio Processing, IEEE Transactions on*, 4(1):31, 1996.