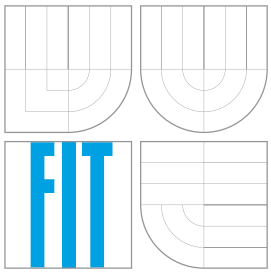
VYSOKÉ UČENI TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DETEKCE ŘEČOVÉ AKTIVITY

VOICE ACTIVITY DETECTION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PETR ENT

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PAVEL MATĚJKA

BRNO 2009

Abstrakt

Práce pojednává o využití support vector machines v detekci řečové aktivity. V první části jsou zkoumány různé druhy příznaků, jejich extrakce a zpracování a je nalezena jejich optimální kombinace, která podává nejlepší výsledky. Druhá část představuje samotný systém pro detekci řečové aktivity a ladění jeho parametrů. Nakonec jsou výsledky porovnány s dvěma dalšími systémy, založenými na odlišných principech. Pro testování a ladění byla použita ERT broadcast news databáze. Porovnání mezi systémy bylo pak provedeno na databázi z NIST06 Rich Test Evaluations.

Abstract

This thesis deals with usage Support Vector Machines (SVM) for Speech Activity Detection (SAD). The first part of the thesis deals with comparison of different feature extractions and different methods of construction supervectors for classifying speech using SVM. The second part presents SVM based SAD system. All experiments were performed on ERT broadcast new database. Final comparison with two other approaches (phoneme and GMM based) was done on standard NIST 2006 Rich Test Evaluation database.

Klíčová slova

detekce řečové aktivity, směs gaussových rozložení, support vector machines, extrakce příznaků, porovnání příznaků, ERT databáze televizního vysílání

Keywords

speech activity detection, voice activity detection, Gaussian Mixture Models, Support Vector Machines, feature extraction, feature comparison, ERT broadcast news database

Citace

Petr Ent: Voice Activity Detection, diplomová práce, Brno, FIT VUT v Brně, 2009

Voice Activity Detection

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Pavla Matějky

.....
Petr Ent
May 25, 2009

Poděkování

Jako první bych chtěl poděkovat Marii Markaki za poskytnutí informací a pomoci v samotných začátcích vývoje systému. Druhé díky míří Pavlu Matějkovi, který mi pomohl posunout systém mnohem dál co se týče odborné a formální stránky.

Contents

Chapter 1

Introduction

Spoken language is all around us. Information is carried in the air to the place of its delivery. It is the same today as the day when the first word was said, but lot of things around has changed since then. In the beginnings, memory played the only role. This way of preserving and spreading information was highly inaccurate, and it was nearly impossible to extract any additional information beside the meaning of the sentence.

And so a man was born. When he grew up, he became a messenger. His job was to memory tribal leader's orders and deliver it to all people in cave.

First milestone was invention of writing. From then on, mankind had a mean of preserving words in time. It also made spreading of information much easier both in time and place. But extra information about the author was still missing.

And so a man was born. When he grew up, he became a scribe. His job was to write down master's orders and post it in the village, to all people read it (ok, those who can read).

Second important moment came with audio recording technique, when suddenly saved spoken language contained many times more information. Apart from the meaning of the words themselves, it was also information about speaker expressed by his voice and information about surrounding conditions. It was possible to store massive amount of data, but another problem rose. How to find one particular information in large audio archives.

And so a man was born. When he grew up, he became a journalist. His job was to attend political meetings and record the speech. Later, he listened to it again couple times, made few notes, rewrote it into a short report and gave it to another man to present it on the TV screen.

Last major step forward was invention of computer and digital media. That is where are we now. We are working hard to remove the problem of previous period. This is when speech processing comes to turn. Its purpose is to extract both original and extra information from the audio to help to find the information of interest in large audio archives. Large Vocabulary Continuous Speech Recognition (LVCSR) provides speech to text transcription, Language Identification (LID) determines language of spoken audio, Speaker Diarization finds number of speakers in recording and marks intervals of activity for each speaker, and many others.

And so a man will be born. When he grows up, he will be a journalist. His job will be to attend political meetings and record the speech. Later, he let diarization system to find speakers in audio, he will label person who he is interested in and let LVCSR make transcription of his speech. He will rewrite it into a short report and he will give it to

another man to present it on the TV screen.

The goal of Speech Activity Detection (SAD) systems (also called voice activity detection systems) is to distinguish parts of audio recording where at least one speaker is active from other parts, such as silence and background noises.

SAD is very useful as front-end for more complex tasks, which could be for example LVCSR, Speaker Diarization, Speaker Identification or LID, where it is important to work only with speech parts of the audio data, because these systems do not take into account long intervals of silence in their input and can behave unpredictable if this situation occurs.

This work shows overview of common techniques used in SAD algorithms, and presents SAD system based on Support Vector Machines (SVM). System was tested on ERT broadcast news database from Computer Science Department of University of Crete and compared with two other systems on NIST RT06s conference meetings database.

This thesis is divided into three main parts, the first is a brief overview of state of the art and used techniques, then follows description of development and testing data and description of tests, and the last part refers to results of the tests.

The goal of this project was to present SAD system based on SVM, which would be comparable with today's SAD systems based on more common principles, such as Gaussian Mixture Models (GMM). This task was successfully completed and developed system produces results which are very competitive.

Chapter 2

Approaches and State of the Art

2.1 State of the Art

Structure of the most of speech activity detection algorithms is similar, two models are trained, one for speech, and the other for the non-speech (silence and background noises). Incoming unknown sample is then classified based on similarity to the models. Decision is usually not sharp, which means that the output is not 1/0 or speech/non-speech, but the probability of belonging to given class. This allows us a lot of variability in representing and post-processing of the results from the classifier. It is also useful when performing speech activity detection as a front-end for the certain task, because demands of different tasks for the input vary. For example Large-Vocabulary Continuous Speech Recognition (LVCSR) works best if the input contains not only the speech, but also small parts of silence near the beginning and the end of the words, meanwhile for example Language Recognition demands just pure speech input.

There are many criterions according to which can be speech activity detection algorithms classified. One of them is type of statistical modeling of the classes used in algorithm. The most common principle is Gaussian Mixture Models (GMM) [5] with maximum likelihood classification algorithm, used for example as a preprocessing stage for Speaker Diarization algorithm developed in TNO. Features from training data are extracted and divided into speech and non-speech classes based on label files. Then two models are trained, one for speech and one for silence. Process of classification starts with feature extraction. Input is then cut into small intervals, which are used to train models, one model per interval. Two options are possible here, either to train model from the beginning, or to use adaptation of existing model. Choice is influenced mainly by amount of the data available for training, since Map Adaptation can work with much less data training of a new model. Trained model is then evaluated and output score is computed. Even though TNO system is mainly oriented to Speaker Diarization, it produces very nice overall error rate of 2.3% in SAD task.

Support Vector Machines (SVM) [1] were introduced in this area recently and they are capable of producing output of the same or even better quality as GMM. One of the first application of SVM in SAD task is described in [8]. Brief introduction to theory of SVM is made and system able to classify audio into 5 different classes is presented. SVMs are capable to distinguish only 2 classes, so more then one evaluation of incoming sample is necessary to classify it. Binary decision tree is used to decrease number of evaluations used per sample to 3. Output error rate for speech/non-speech classification is 3.35%.

These are the main ideas in this field, however there are few systems with completely

different approach. One of them is usage of phoneme recognizer to detect isolated phonemes in audio recording. Found phonemes are replaced with speech labels and basic classification is done. Results later needs some form of smoothing, because there are small gaps around each detected phoneme, which degrades system results. Basic ideas how this system works and comparison with system based on SVM are described in section 5.3.

2.2 Feature Extraction

Virtually all algorithms working with speech data first converts time-domain speech signal to feature representation. This conversion tries to transform signal into more compact form while preserving information which it carries. First, short term spectral analysis is performed to transform signal from time domain to frequency domain. Result is then passed to a filter, which modifies the information according to a human model of hearing. Two most common methods are Mel-frequency Cepstral Coefficients (MFCC) analysis and Perceptual Linear Predictive (PLP) analysis. In short note, MFCC transformation first applies mel-bank of filters and takes logarithms of its amplitudes. MFCCs are then output of Discrete Cosine Transformation (DCT), which has these values as its input. Special position has the 0^{th} coefficient, which is lightly modified sum of log filter bank amplitudes. MFCC extraction algorithm is described in depth in [5]. PLP analysis first also applies mel-bank of filters and takes logarithms of its amplitudes. These coefficients are weighted by an equal-loudness curve and then compressed by taking the cubic root. From resulting spectrum, Linear Prediction Coefficients (LPC) are estimated. DCT is finally used to compute output PLP coefficients. In depth descriptions is provided in [4]. Delta and acceleration coefficients are often computed from basic features. Deltas are computed using Equation 2.1

$$\Delta_t = \frac{\sum_n^{w_\delta} n(c_{t+n} - c_{t-n})}{2\sum_n^{w_\delta} n^2} \tag{2.1}$$

where Δ_t is delta coefficient at time t, w_Δ is window size for delta calculation and c_x is coefficient at time x. Acceleration coefficients are computed using the same formula, except that c_x is replaced by Δ_x and window length w_Δ is replaced by w_α. In HTK toolkit, $w_\Delta = 2$ and $w_\alpha = 2$.

Another feature, which is commonly used for SAD task, is energy of the signal. It is computed using Equation 2.2.

$$E = log \sum_n^N s_n^2 \tag{2.2}$$

where E is output energy, N is number of samples of the signal and s_n is the n^{th} sample of the signal. It is very similar feature to 0^th MFCC coefficient, as they are both computed using the same operations.

Lot of algorithm uses also zero crossing rate of the signal. It is the number of times when signal changes its sign on given interval and is computed using equation 2.3.

$$r_c = \sum_n^N (s_n * s_{n+1} < 0) \tag{2.3}$$

where s_n is the n^{th} sample from input interval and r_c is zero crossing rate for this interval.

2.3 Gaussian Mixture Models

2.3.1 Structure of Gaussian Mixture Models

Gaussian Mixture Models (GMM) are often used as probability distribution functions in SAD algorithms. They are modeled out of a mixture of weighted Gaussian functions. Gaussian function is defined by its mean vector μ and covariance matrix Σ. If input data are not correlated, covariance matrix is from computational purposes often reduced to diagonal form, which results in vector of variances σ. Formula 2.4 define how in output likelihood computed for N Gaussian mixture with μ and σ.

$$l(x) = \sum_{n}^{N} w_n \mathcal{N}(x, \mu_n, \sigma_n) = \prod_{d}^{D} \frac{1}{\sigma_{nd}\sqrt{2\pi}} e^{-\frac{(x-\mu_{nd})^2}{2\sigma_{nd}^2}} \tag{2.4}$$

where l_n is output likelihood, w_n weight of the n^{th} Gaussian function in mixture, x observation sample and D number of data dimensions. Before GMM can be used in speech applications, it has to be trained, which means adjusting μ and σ according to the input data. Training is performed in cycles of two phases. First phase trains existing model. Expectation Maximization (EM) algorithm is probably the most commonly used for this purpose. This algorithm is run in iterations and in each iteration, it adjusts values of μ and σ so that output likelihood is greater or equal to the likelihood from previous iteration. However EM does not grant to find optimal solution. Equations 2.5, 2.6, 2.7 and 2.8 are used for GMM parameters re-estimation.

$$\gamma_i(s) = \frac{w_i \mathcal{N}(x_s, \mu_i, \sigma_i)}{\sum_n^N w_n \mathcal{N}(x_s, \mu_n, \sigma_n)} \tag{2.5}$$

$$w_i = \frac{1}{N} \sum_{s}^{S} \gamma_i(s) \tag{2.6}$$

$$\mu_i = \frac{\sum_s^S \gamma_i(s) x_s}{\sum_s^S \gamma_i(s)} \tag{2.7}$$

$$\sigma_i = \frac{\sum_s^S \gamma_i(s)(x_s - \mu_i)^2}{\sum_s^S \gamma_i(s)} \tag{2.8}$$

where $\gamma_i(s)$ is occupation function for the i^{th} Gaussian distribution and the s^{th} training vector, x stands for training data and S is total number of training vectors. EM algorithm just trains existing GMM, however does not modify the number of distributions in the model. So in second phase, new Gaussian function is added to the model. Gaussian function with the biggest weight w is chosen to be split. New variances σ_1, σ_2 remain the same as original σ, weights $w_1 = w_2 = w/2$ and means $\mu_1 = \mu + 0.2\sigma$ and $\mu_2 = \mu - 0.2\sigma$.

Building of a more complex model requires relatively large amount of training data. When not enough training data for given class is available, solution is to adapt universal model trained on data from more classes. Such model is called Universal Background Model (UBM) and commonly used technique of adaptation is MAP Adaptation. It modifies μ and σ of UBM based on given observations. Equations of the modification for model with m Gaussian distributions are 2.9, 2.10 and 2.11.

$$N_m = \sum_{s}^{S} \mathcal{N}(x_s, \mu_m, \sigma_m) \tag{2.9}$$

$$\mu_{mo} = \frac{\sum_{s}^{S} \mathcal{N}(x_s, \mu_m, \sigma_m) x_s}{\mathcal{N}(x_s, \mu_m, \sigma_m)} \tag{2.10}$$

$$\mu_{mn} = \frac{N_m}{N_m + \tau} \mu_m o + \frac{\tau}{N_m + \tau} \mu_m \tag{2.11}$$

where N_m is occupation likelihood of adaptation data, μ_{mo} is mean of adaptation data for the m^{th} mixture and μ_{mn} is output mean. This is the simplest case, where variances are not changed. In depth description of more complex cases can be found in [5].

2.3.2 Viterbi

A special case of usage of the Viterbi algorithm is described in this chapter, followed by explanation of one of its implementations, token passing algorithm. The Viterbi algorithm in general is nicely described in [5]. Version of the Viterbi algorithm used in SAD works with model containing just two states, one for speech, one for non-speech. Its input are probabilities from classificator, output is vector of states. Its purpose is to find optimal sequence of states, based on input probabilities and given transition matrix. Transition matrix contains transition probabilities which along with the input sample determine whether to stay in current state, or switch to the other one. A convenient method of Viterbi implementation is token passing algorithm. In this algorithm, each state is able to hold a so called token, which contains optimum probability of being in particular state at time t_p and vector of passed states. At time t_0, both states are initialized by token with empty path vector and probability 1. With each input value, tokens from both states are split to two identical and modified using Equation 2.12

$$s_{t+1} = s_t + P(x, y)i \tag{2.12}$$

where s_t is probability saved in token at time t, $P(x, y)$ is transition probability from state x to state y, and i is input value. One of the tokens uses probability $P(x, x)$ - staying in current state, while the other uses $P(x, y)$ - switching the state. After each modification, two tokens arrive in the same state, as shown in Figure 2.1.

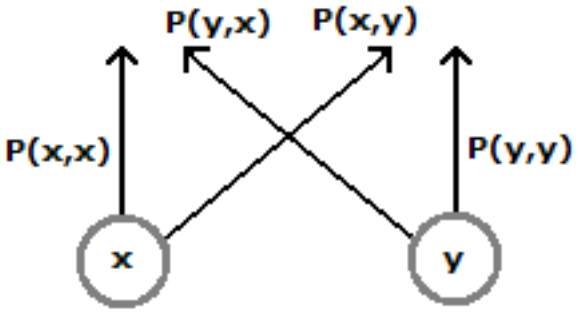


Figure 2.1: Figure shows one step of token passing algorithm.

Token with higher probability is chosen, current state is added to its path vector and the other token is thrown away. In the end, token with higher probability is chosen, and its path vector represents the optimal order of states.

2.4 Feature Modification

Process of feature modification has three parts. First, interval of certain number of frames is cut out from feature file. Then one of the modifications is applied on this interval. This is repeated for all desired intervals. As the last step, variance of each component of all resulting vectors is computed, creating vector of variances. Each of the resulting vectors is then divided by the variance vector, component by component. Variances are saved and used later to divide components of validation and test vectors. This procedure is done to unify dynamical range of feature vectors components in all three data sets.

2.4.1 Mean and variance

The first and easiest method is to take mean and variance over certain number of consecutive, as shown in Figure 2.2.

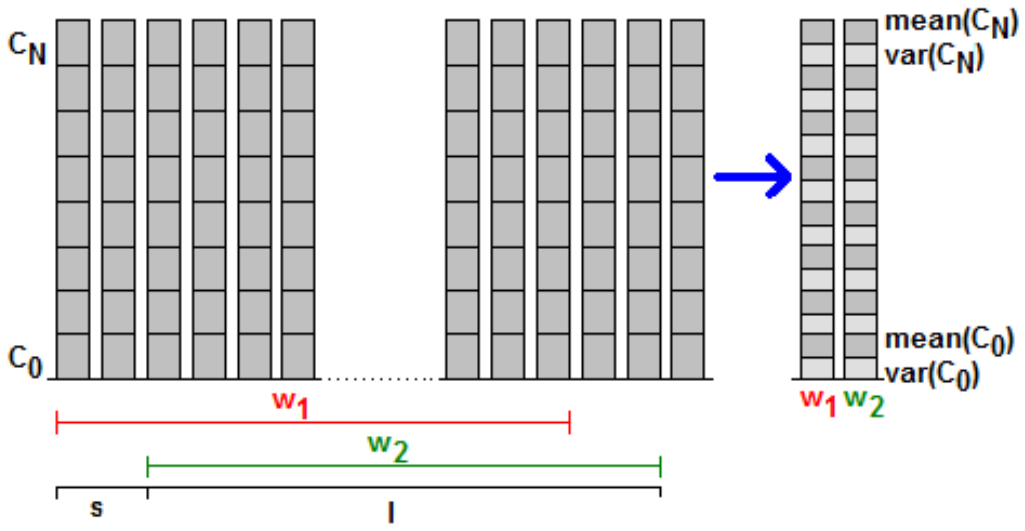


Figure 2.2: The transition from feature vectors to training vectors for SVM, using mean and variance of input features. C_x from feature vector can refer to one of the MFCC coefficients, energy or zero crossing rate. w_x denotes the x^{th} window, resp training vector, l length of window and s shift of this window.

This method was proposed in [9]. It demands very few computational resources. Very similar to this method is median and variance modification. It uses the same process, except that mean is replaced with median. Number of members in output vector is twice the number of members in input vector for both these methods.

2.4.2 Discrete Cosine Transform

Another possible modification method is usage of first n Discrete Cosine Transform (DCT) coefficients. Transition from feature vector to training vectors using DCT is shown in Figure 2.3.

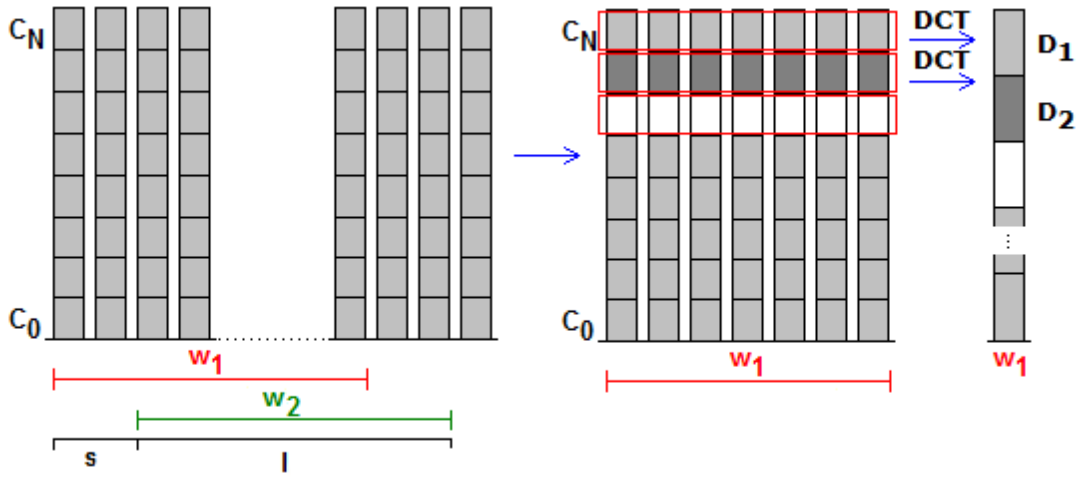


Figure 2.3: The transition from feature vectors to training vectors for SVM, using DCT coefficients of input features. C_x from feature vector can refer to one of the extracted coefficients, energy or zero crossing rate. w_x denotes the x^{th} window, resp training vector, l length of window and s shift of this window. DCT denotes discrete cosine transform operation and D_x sets of first n DCT coefficients.

Input of all modification methods is p feature vectors. Value of p has to be chosen carefully, as some operations need longer window times to work correctly. For example DCT coefficients computed from just a few points cannot provide any useful information for vector comparison. This is the main limitation of DCT method, because other methods can successfully work with almost any number of input vectors.

2.4.3 Gaussian Mixtures

Computation of mean and variance described above is the same as one iteration of single mixture GMM training algorithm. Therefore one of the ideas was to determine influence of adding more Gaussians to the GMM model.

Starting Gaussian, represented by means and variances, is trained using expectation maximization algorithm and split into two new Gaussians. This process is repeated until desired number of components is reached. Means and variances of these final Gaussians are then concatenated into one vector and used for SVM model training in the same way as parameters from one Gaussian. This process is shown in Figure 2.4

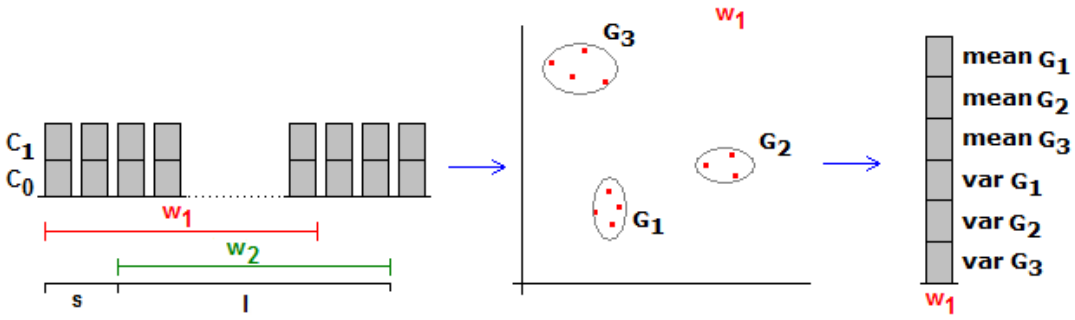


Figure 2.4: The transition from feature vectors to training vectors for SVM, using means and variances from multiple Gaussians. For simplification, only two dimensional input features and three Gaussians are used. C_x from feature vector can refer to one of the MFCC coefficients, energy or zero crossing rate. w_x denotes the x^{th} window, resp training vector, l length of window and s shift of this window. $mean_x$ denotes means of the x^{th} Gaussian and var_x its variances.

2.5 Support Vector Machines

Support Vector Machine is an algorithm, which creates an optimal separating hyperplane (also called maximum margin hyperplane) from provided positive and negative train samples. This is realized by creating two margins, which are as far as possible from each other while still separate the data points. Separating hyperplane is defined as geometrical center between these two margins, as is shown in Figure 2.5.

Formally written, set of input samples is defined as

$$\mathcal{D} = \{(x_i, c_i)|x_i \in \mathbb{R}^p, c_i \in \{-1, 1\}_{(i=1)}^n \tag{2.13}$$

where x_i denotes p dimensional real vector and c_i class to which it belongs. Algorithm is trying to construct hyperplane, which divides points with $c = 1$ from those with $c = -1$. Any hyperplane is described by set of points x satisfying Equation 2.14.

$$w \cdot x - b = 0 \tag{2.14}$$

where $\cdot$ means dot product and b distance of the hyperplane from the origin. Vector w is normal vector perpendicular to the hyperplane. Margins are defined in Equations 2.15 and 2.16.

$$w \cdot x - b = 1 \tag{2.15}$$

$$w \cdot x - b = -1 \tag{2.16}$$

Task of the training algorithm is to place these two parallel hyperplanes as far as possible from each other while still separating the data. At this point, training can be influenced

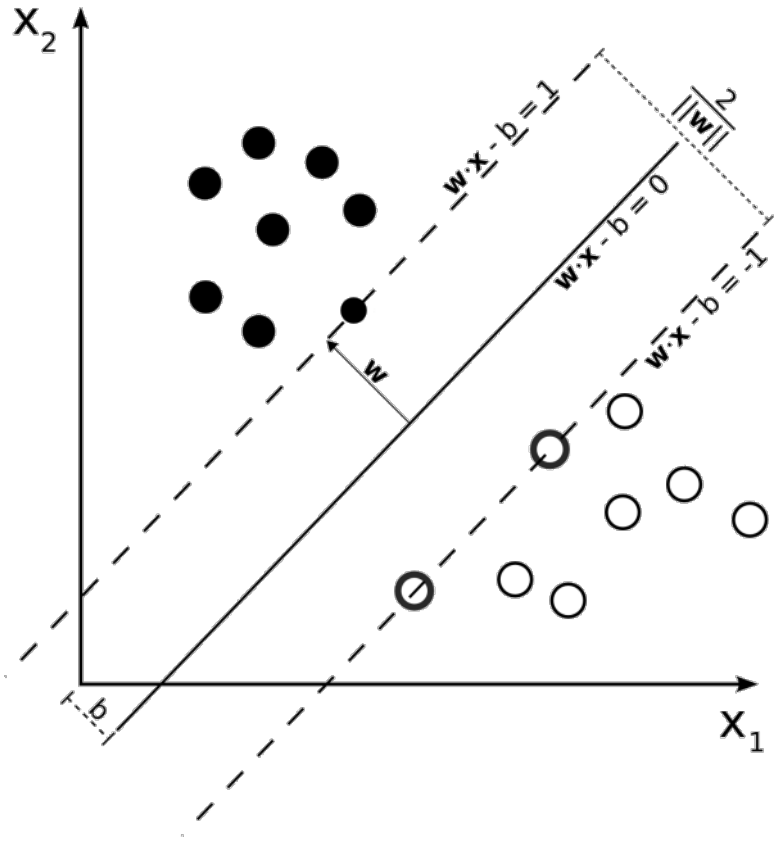


Figure 2.5: Separating hyperplane for two dimensional data (solid line), two margins (dashed lines) and samples from two classes. [14]

by setting variable *err*, which defines maximal ratio of misclassified samples. Consider the situation in Figure 2.6.

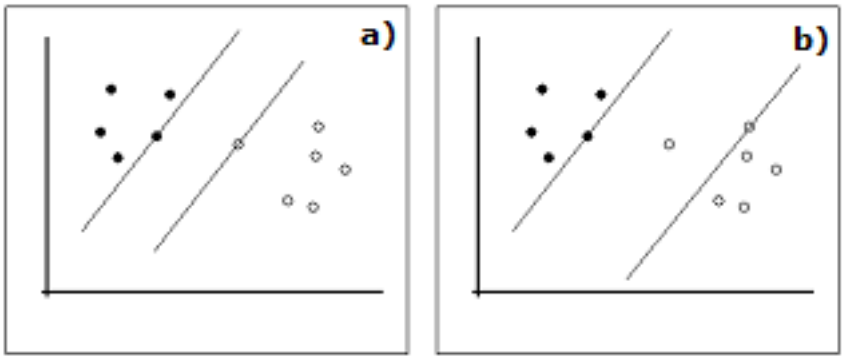


Figure 2.6: a) Margins in training phase with no errors allowed. b) Margins in training phase with *err* errors allowed.

When one of the samples is away from the rest of the data and *err* is set to 0, algorithm

still has to count with this sample and margins in resulting model are close to each other. However if this point is sacrificed, margins are much further from each other and model has potential to outperform the first one.

By transforming Equations 2.15 and 2.16, the distance between them is defined as $\frac{2}{\|w\|}$. Therefore the task of the training algorithm is to solve optimization problem

Minimize

$\|w\|$

subject to (for i=1,...,n)

$c_i(w \cdot x - b) \geq 1$

for at least $(1 - err)n$ from n points.

Points of both classes on the margins are called support vectors and their geometrical distance from separating hyperplane is $d_m = \frac{1}{\|w\|}$. When classifying unknown vector, its score is computed using following equation:

$$s = \frac{d}{d_m}$$

where s is output score, which could be viewed as a probability of belonging to given class, and d is distance of classified vector from separating hyperplane.

Support vector machines are either linear, or non-linear with kernel function. In some cases, training data are so complex, that it is impossible to construct separating hyperplane in original space. Kernel based systems solve this problem by transforming data into another space, where input data points are linearly separable. Transforming function can be non-linear and transformed space can have more dimensions than the original one. Nonlinear transformation is shown in Figure 2.7.

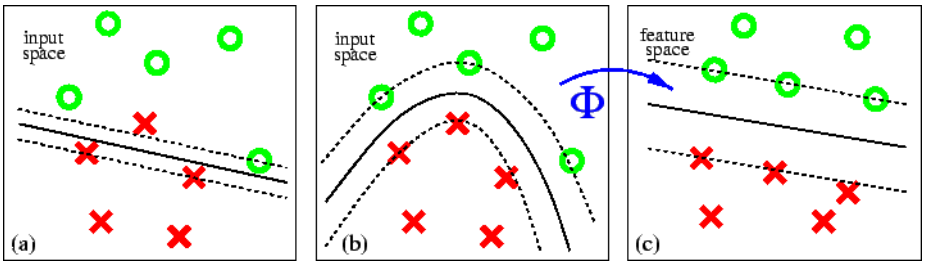


Figure 2.7: a) linear separation of two classes with miss-classifications b) optimal non-linear separating function is found in input space c) input space is transformed by found function Φ, linear separation is achievable [2]

Chapter 3

SAD system description

The algorithm can be divided into two main parts, training and evaluation phase.

3.1 Training phase

The goal of the training phase is to build model for SVM classifier from training part of the database. This phase is run only once, created model and vector component variances are saved and used in all future evaluations. Algorithm of model building is described in following steps:

1. extraction of speech and non-speech time intervals from file transcriptions
2. feature extraction
3. feature modification
4. SVM/GMM model training

Feature extraction is done via one of the tools described in section 5.1. Time intervals are then extracted from label files. Only intervals longer then $l_i ms$ are taken. l_i is defined as

$$l_i = n * l_f$$

where n is number of frames over which is mean and variance computed later and l_f is length of one frame. Then intervals of exact length l_i are made, defined as

$$< 0, l_i >< l_i, 2 * l_i > ... < l_s - l_i, l_s >$$

where l_s is length of original interval extracted from label file. At this point, one of the feature modifications is applied (mean and variances, median and variances, multiple Gaussians or DCT). The last step of this phase is training of the model. Setup of SVMLite training algorithm was taken from [9] and is following

t	=	2	radial basis kernel
g	=	0.01	γ parameter in kernel
c	=	1	tradeoff between training error and margin

3.2 Evaluation phase

The goal of the evaluation phase is clear, to make the best possible speech non-speech segmentation. Steps to do this are following

1. feature extraction
2. time intervals creation
3. feature modification
4. SVM classification using pretrained model
5. result interpretation and post-processing

First step is the same as in the training phase. Time intervals are then created as follows

$$< 0, l_i >< s_e, l_i + s_e > ... < l_f - l_i, l_f >$$

where l_i is length of the window for feature modification, s_e is shift of this window and l_f is length of the file. Interval length l_i must be the same as length of intervals used for model training. Different values of l_i and s_e were tested to get the best system performance. The same feature modification as in training phase is then applied. Resulting feature vectors are divided by saved variances from training phase and classified by SVM, giving probability of belonging to one of the classes. These numbers can be presented as vector of states, where positive values means speech and negative non-speech, but a lot of misclassification errors occur. Therefore one of the following smoothing algorithms is applied. First algorithm is proposed by [8] and has form of rule, which is applied on vector of binary states. States are generated from SVM output probabilities using following conditions

$$\begin{aligned} v_t > th- > s_t &= 1 \\ v_t <= th- > s_t &= 0 \end{aligned}$$

where v_t is probability of belonging to speech/non-speech class in time t, th is threshold and s_t is output value in time t. Value th is key for system output and it is the threshold used for DET curve plotting.

General rule of the algorithm is

$$if\ s_{t-1} == s_{t+1}\ and\ s_t! = s_{t-1}\ then\ s_t = s_{t-1}$$

where s_x represents state in time x. Expressed in words, it is filling of one frame holes in sequence. This rule removes very short intervals of silence and makes result more consistent.

As a second option, viterbi decoder can be used. It is applied on probabilities of belonging to speech/non-speech class. Model for viterbi decoder is constructed from two states, speech and non-speech. Transition probabilities are sp_s/ns_s for staying in speech/non-speech state, and sp_c/ns_c for changing the state. Basic idea is that if transition probabilities are set that $sp_s > sp_c$ and $ns_s > ns_c$, system tends to more likely keep the current state even if input probability belongs to opposite state. This removes short silence and speech intervals.

Chapter 4

Experimental setup

4.1 Training, validation and testing data

4.1.1 ERT Broadcast News database

System was developed using ERT Broadcast news database from Computer Science Department of University of Crete [9]. Whole database contains about 26 hours of labeled data. Three subsets were made, training and testing contained 5.5 hours of audio, validation one contained 3 hours.

Database contains standard broadcast news - speech, silence, noise, music and commercials, but the developed system is meant to work with meeting data, where only speech, silence and noise occur. Therefore start and end times of commercials and signature of TV station are saved from input label files and later excluded from system output and performance measurements. As the result, database which simulates content of standard meeting data is obtained. This modification had to be made because no meeting data database with labels and transcriptions was available at the time of system development.

4.1.2 NIST RT06s Conference Meetings database

Database from RT06s conference meetings is used for comparison with SHoUT SAD system [7]. It contains recordings of 27 meetings, from which 8 is used for system evaluation and the rest is for development. Whole duration of the files is not used during evaluation, there is script defining intervals of interest for each file. Total time used for evaluation is about 2 hours.

4.2 Confidence measurements

Performance measure presented in NIST 2005 and 2006 speech activity detection benchmarks is used to evaluate system. It uses fact that output of SAD system can be considered as a special case of speaker diarization system result. SAD is a segmentation task with two classes - speech and non-speech. So if all speakers from reference label files are joined into one class, these new label files can be used as a reference for speech activity detection. Result of the system is scored by speaker diarization evaluation script used in Rich Transcription Evaluation 2006 [10]. Output error rates are defined as follows

$$FA = \frac{t_f}{t_s}$$
$$MISS = \frac{t_m}{t_s}$$

where FA is false alarm rate, t_f is false alarm time (time when silence was falsely classified as speech), t_s is total speech time in recording, $MISS$ is miss rate and t_m is miss time (time when speech was not classified as speech). Overall detection error rate is defined as

$$DER = FA + MISS$$

and it is the percentage of falsely classified speech time.

If some of the parameters of the system can be tuned to achieve optimal performance, detection error curve (DET) can be plotted. It describes dependency of miss rate on false alarm rate when changing value of tuned parameter. DETware MATLAB toolbox [11] is used to compute and to plot DET curves. In first set of tests, which are focused on finding optimal feature set, tuned parameter is threshold applied to output probability when converting it to binary form. Example of the situation is shown in Figure 4.1. Various variables are used to plot DET curves in second phase, when the goal is to find optimal setup of the system.

When plotting DET curve, important point is equal error rate (EER). It is read out from the plot at the point where FA and MISS are equal and it is defined as

$$EER = \frac{FA + MISS}{2}$$

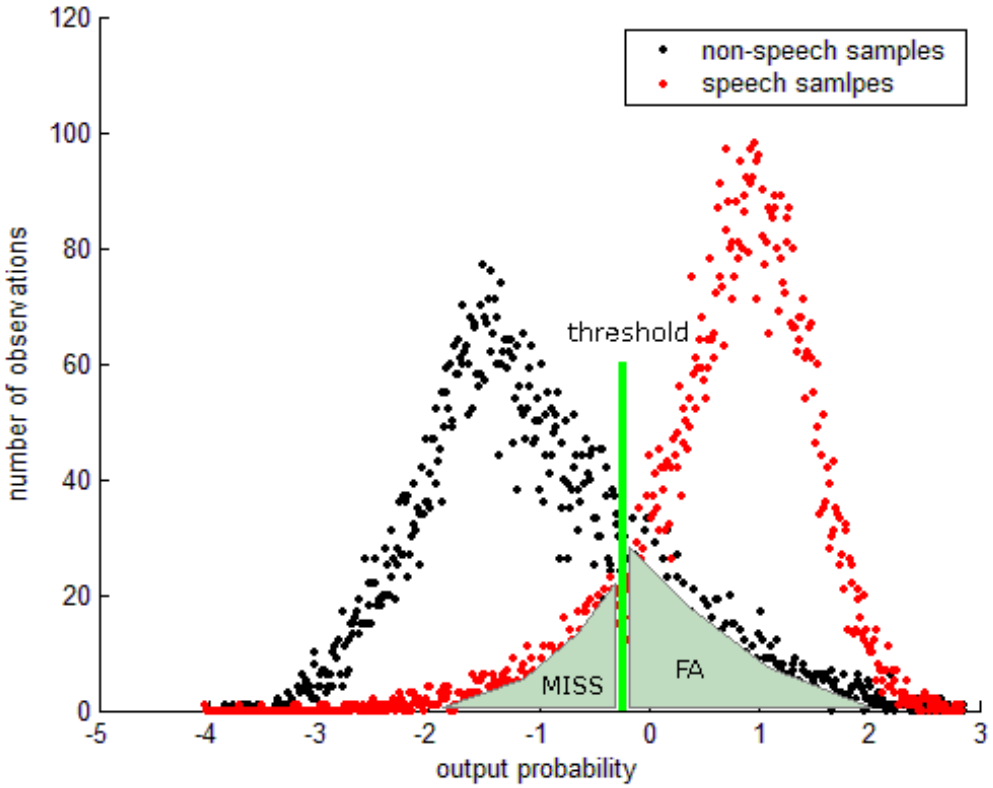


Figure 4.1: Histogram of output probabilities in validation phase. Red dots on the right side are speech samples, black dots on the left are non-speech samples. Any point, which is known to be speech, but is on the left from the threshold, is called a miss. Non-speech points above the threshold are called false alarms.

Chapter 5

Results

5.1 Comparison of different feature sets

5.1.1 Experiments description

One of the tasks for this thesis is to determine the best combination of features and modifications for SVM based SAD system. The first part of the tests is concentrated to this task.

The very first test was focused on determining the optimal about amount of training vectors needed for proper model training. Its results were used to divide database into training, validation and testing subsets, which were used in later tests, concerning features and modification methods.

Choice of type and number of extracted features influences a lot the final results of the system. Determining the best type of the coefficients for speech activity detection was one of the goals of system development. First, tool for feature extraction had to be chosen. Candidates were feacalc from SPRACHCore library [13], which is built on rasta library, and HCopy tool from HTK toolkit [6]. Regarding the type of extracted coefficients, literature proposes two candidates, Mel Frequency Cepstral Coefficients (MFCC) and Perceptual Linear Prediction coefficients (PLP).

This comparison is made with first 12 MFCC coefficients, as it produces the best results according to comparison in [3]. Number of PLP coefficients is also 12, to have direct comparison of the same number of different coefficients. MFCC are computed using hamming window, 19 filter banks, and they are extracted from frames of length 25ms with 15ms overlap. PLP coefficients are extracted using same configuration as MFCC plus the number of intermediate stage Linear Prediction Coefficients (LPC) is set to 12.

Influence of delta (Δ) and acceleration coefficients ($\Delta\Delta$), energy, the 0^{th} coefficient and zero crossing is also part of the tests.

In the end, various feature modifications are compared.

Quality of the final result can be influenced by exact sequence of choices, which should be ideally made parallel. Examples are choice of extraction tool, type of coefficients, delta and acceleration coefficients, method for coefficients modification and so on. In ideal case, all of the possible combinations should be tested to be sure that optimal combination is found. This attitude would result to

$$c = \prod_{i}^{n} x_i$$

where c is number of possible combinations, n number of decision and x_n number of choices for the n^{th} decision. In actual numbers, 9 decisions has to be made, the most of them with two choices, which results in more then 512 possible combinations. Test of one combination requires feature extraction and model training and evaluation, which takes together about 45 minutes. Exploring all possible solution space would mean enormous computational load, so another way has to be used.

Decisions are made one by one, from the most general one to the most specific one. After one decision is made, all following tests are performed with features extracted according to this decision. Default values of some parameters have to be chosen, as their setup is part of later tests, but they are used from the beginning. The default setup is 13 ($0^{th} - 12^{th}$) coefficients with mean + variances as method for feature modification. With this attitude, there is a possibility that the optimal combination is not found, but it is lowered by right choice of tests order. Problem of the computational load is solved, because now the number of test required is a sum of the choices and not a multiple of choices as in first case.

The order of tests is set as follows

1. Feature type
2. Tool for feature extraction
3. Energy and the 0^{th} coefficient
4. Zero crossing rate
5. Delta and acceleration coefficients
6. Method of feature modification

All the tests described above are made to determine the best combination of features in the feature vector. Model is trained on the vectors from training set and then it is tested on vectors from validation set. Testing vectors are either pure speech or pure non-speech and this type is known at the time of their evaluation. The result of these tests is accuracy of recognition of test vectors by given model.

As the result of this phase, the best combination of features, modification method and feature extracting tool is chosen and speech activity detection system is built around them.

5.1.2 Training data size

Test is made with following features: the first 13 PLP coefficients and energy are extracted and mean + variance modification method is applied. The first task is to find dependency of model complexity (vectors used as support vectors) on number of frames passed to modification method. This is necessary to find out which model needs the largest amount of training data, so next tests can work with the this model as with worst case. Models for this task were trained on 20000 vector of each speech and non-speech class. Table 5.1 shows observed results.

Model trained on vectors made from 5 frames is the most complex. The cause for that is large variability in training vectors, since the means and variances are made only from 5 values. As the number of frames passed to modification method rises, values in vectors become more stable and models are less complex. Next test is therefore made with 5 frame model, as it is the most complex one and needs the largest amount of training vectors to

Number of frames passed to modification method	Number of support vectors in model
5	6095
10	4374
15	3502
20	2991
25	2648

Table 5.1: Listed values shows dependency of support vectors in model on number of frames passed to modification method. The higher number, the more complex model.

construct optimal separating hyperplane. 25 frame model is added to graph to prove that behavior of the system is not dependent on number of frames used to create training vector, but only on number of this vectors.

Amount of training vectors varies from 1000 to 40000 for each class, trained model is validated with 20000 vectors from validation set (10000 each class). Equal error rate is found using threshold for transition from output probabilities to speech/non-speech states and graph shown in Figure 5.1 is made from these results.

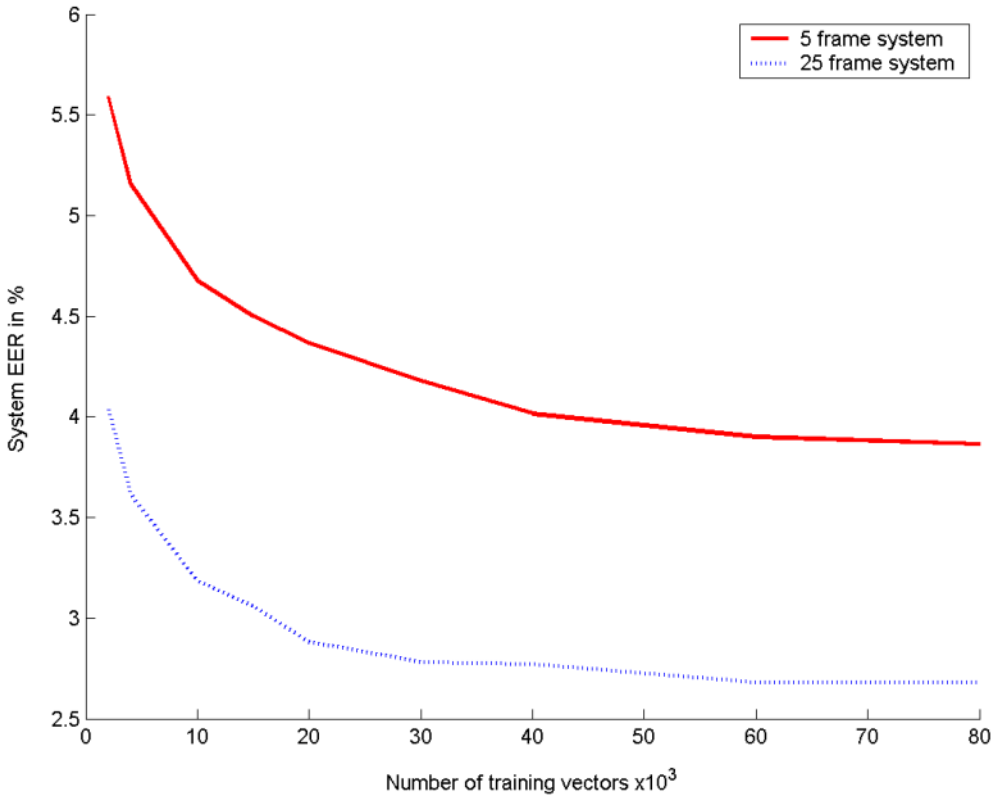


Figure 5.1: Graph of dependency of EER on number of training samples. Red solid line is for model trained on vectors made from 5 frames, blue dotted line for training vectors from 25 frames.

Graph shows that the more training data, the better classification is achieved. It is also proven that number of frames used for vector computation does not play any role in

this test, because both curves have very similar shape. Noticeable think is that model trained on 25 frame segments provides better classification (lower EER). Reason is that mean computed from 25 frames is very similar across all the training vectors of the same class and it is much easier for SVM to classify it.

Time necessary for model training grew approximately linearly with number of training vectors, but improvement in output EER has nearly exponential dependency. Because many tests had to be made, 20000 training vectors for each class was chosen as compromise between time of model training and system performance. Adding more training data to the system does not significantly improve system performance while just extends computational time.

Size of training vectors for different models is up to 250ms, so training data size t has to be at least

$$t = 20000 * 0.25s = 5000s$$

for each class. Speech to non-speech ration in ERT database is approximately 3:1, so to have $5000s$ of silence, $15000s$ of speech have to be taken. In total, training and testing set contains $20000s \doteq 6h$ of data. Validation phase works with 10000 vectors for each class, as this number is big enough to prevent statistical errors. This means that half the size of training set, 3 hours, is taken.

5.1.3 Feature type

First test is concentrated on comparison of PLP and MFCC coefficients. Set of first 13 coefficients (including the 0^{th} one) of each kind is used for this purpose. Configuration parameters for feature extraction are set maximally similarly - number of filter bands, preprocessing and post-processing - to ensure that the difference of the results will be caused mostly by nature of coefficients itself. HTK's HCopy is used for feature extraction. Figure 5.2 shows the results.

There is no measurable difference in EER values, as shows Table 5.2

	EER
MFCC	3.75%
PLP	3.75%

Table 5.2: Comparison of equal error rate (EER) of PLP and MFCC from first test.

PLP coefficients are chosen as better option, because they outperforms or have at least the same result as MFCC on the major part of the plot.

5.1.4 Tool for feature extraction

Second test is about to determine which tool is better for PLP extraction. Candidates are feacalc from SPRACHCore library and HCopy from HTK toolkit. As process of extracting PLP coefficients is well defined in literature and the same configuration is used, is was expected to obtain very similar results, where differences could be caused only by few factors - different shape and position of mel-filters or their adjustment by any other means. Results are shown in Figure 5.3.

Values of EER are listed in Table 5.3.

HCopy is chosen as a tool for feature extraction, because it provides a little better results and it also offers control over more parameters during extraction process. Values of EER

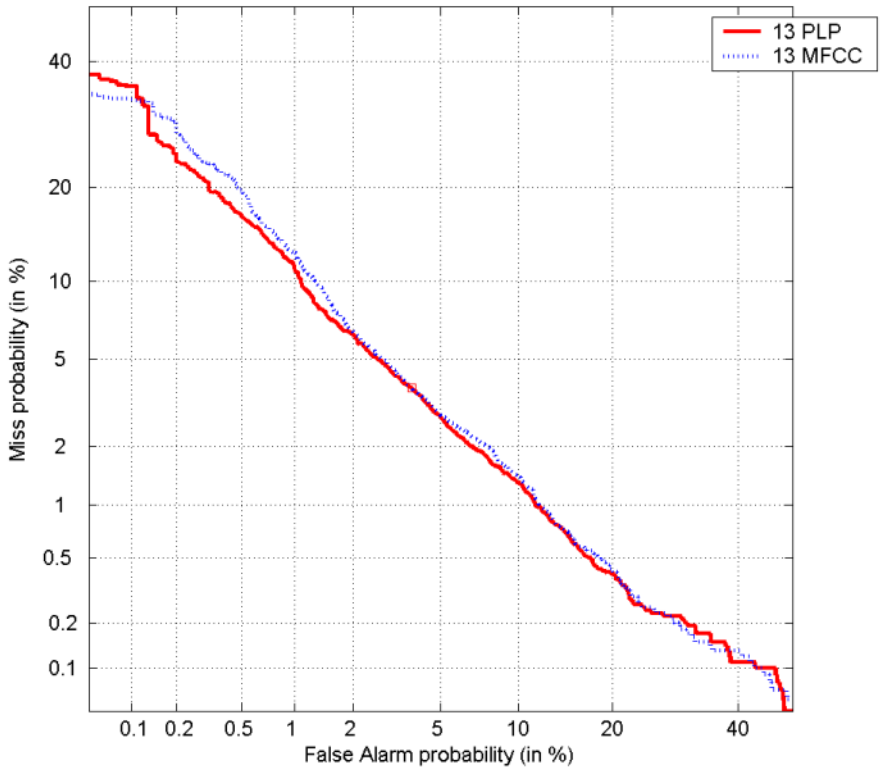


Figure 5.2: DET curve for PLP (red solid line), MFCC (blue dotted line) and equal error rate points.

	EER
HTK	3.85%
feacalc	3.97%

Table 5.3: Comparison of equal error rate (EER) of HCopy's and feacalc's PLPs.

are different from previous test, because feacalc by default computes PLP coefficients from the 1^{st} one and has only option to add energy, but not the 0^{th} coefficient, so configuration file of HCopy had to be adjusted to match this settings.

5.1.5 Energy and the 0^{th}oefficient

Next test explores influence of the 0^{th} PLP coefficient and energy on system performance. First, error rate of system with 12 PLP coefficients starting from the 1^{st} is measured. Then effect of adding energy, the 0^{th} coefficient or both is explored. Results are listed in Table 5.4.

System produces the best results when both energy and the 0^{th} PLP coefficient are added. It also shows that adding the 0^{th} coefficient improves system output little more then adding energy. If neither energy nor the 0^{th} coefficient is extracted, system performance degrades about 37% compared to system with at least one of them.

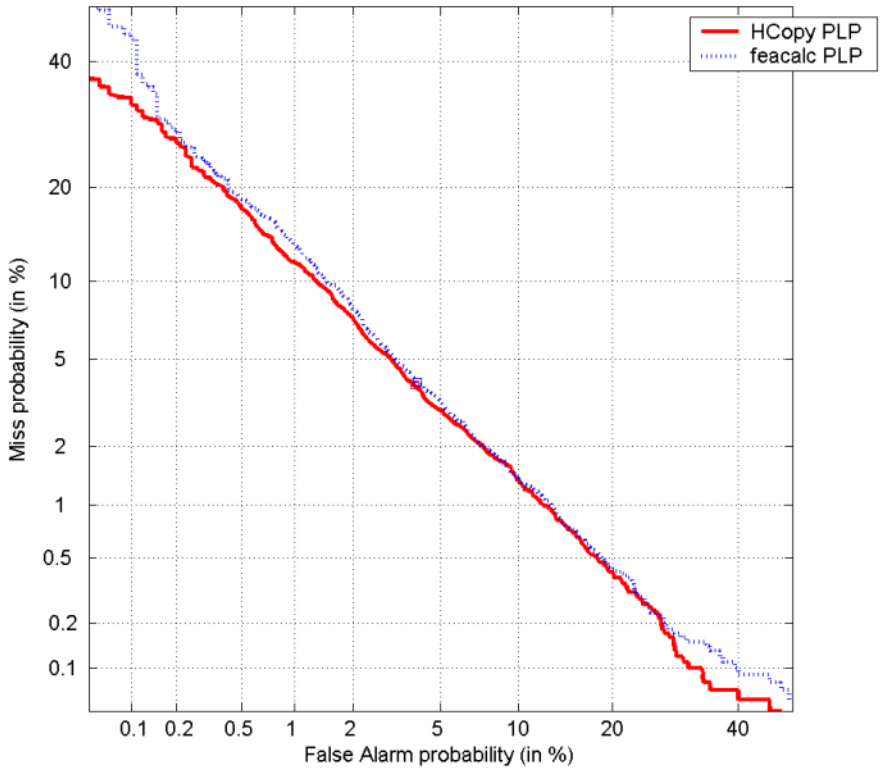


Figure 5.3: DET curve for 12 PLP coefficients and energy extracted by HCopy (red solid line), feacalc (blue dotted line) and their EER points.

	EER
12 PLPs	5.14%
12 PLPs + energy	3.86%
12 PLPs + 0^{th}	3.75%
12 PLPs + energy + 0^{th}	3.66%

Table 5.4: Comparison of equal error rate (EER) and of 12 PLP coefficients alone, with energy, with the 0^{th} coefficient and with both.

5.1.6 Zero crossing rate

Fourth test shows system performance if zero crossing rate is added as another feature. Error values are listed in Table 5.5. Zero crossing rate is computed on the same intervals as PLP coefficients are and it assumes that input files have no DC offset, which is fulfilled in ERT database.

Zero crossing rate is commonly used and it is concerned as useful feature for SAD. Results in Table 5.5 are therefore very surprising. At first, noise seemed to be the reason for such a bad results, because it has a lot of high frequencies which may corrupt zero crossing rate. So in next test, low pass filter was used as preprocessing. Low pass filter length was 64 coefficients and only 15% of the lowest frequencies passed. The goal was

	EER
12 PLPs + 0^{th} + $energy$	3.66%
12 PLPs + 0^{th} + $energy$ + zcr	3.69%
12 PLPs + 0^{th} + $energy$ + $zcr, preprocessing$	3.67%

Table 5.5: Comparison of equal error rate (EER) of 12 PLP coefficients, 12 PLP coefficients augmented with zero crossing rate and influence of preprocessing.

to remove noise from input. But the result does not show any noticeable improvement compared to system without zero crossing rate.

5.1.7 Delta and acceleration coefficients

Next test is aimed to explore system performance when adding delta and acceleration coefficients. Configuration of feature extraction phase, based on previous tests, is following: $0^{th} - 12^{th}$ PLP coefficients + energy, extracted by HCopy. Modification method is mean + variances. Values of EER are listed in Table 5.6.

	EER
plain system	3.66%
added Δ	3.38%
added Δ + $\Delta\Delta$	3.19%

Table 5.6: Comparison of equal error rate (EER) of plain system, system with deltas and with delta and acceleration coefficients.

Delta and acceleration coefficients improves together system performance by 15% and are used in all following tests.

5.1.8 Method of feature modification

In the last test in this section, 3 modification methods are tested to replace mean + variances. It is median + variances, discrete cosine transform (DCT) and multiple Gaussians. EER from comparison of mean and median are in Table 5.7.

	EER
mean + variances	3.66%
median + variances	3.72%

Table 5.7: Comparison of equal error rate (EER) of system using mean + variances with system using median + variances modification method.

Median modification was slightly worse. Comparison of mean with other two methods was a a little difficult. First problem was with number of input values. For DCT, number of input values means length of virtual input signal, for multiple Gaussians number of points to place Gaussians on. Since testing is performed on 15 frames (15 inputs), observations had to be made if methods could work with this size. There is no problem with DCT, but training of multiple Gaussians is correct only for 2 mixtures in model. When training three and more mixtures, only two unique sets of means and variances appears in the final model. Number of mixtures is therefore limited to 2. Second problem concerns usage of delta and

acceleration coefficients along with DCT or multiple Gaussians. New modification methods improves system behavior when plain (without Δ and $\Delta\Delta$) features are used, but when Δ and $\Delta\Delta$ are extracted, results are worse then without them. The possible reason can be that Δ and $\Delta\Delta$ are computed from features, so they can be considered as some kind of feature modification, same as for example DCT. It is possible that if another non-trivial modification (DCT, multiple Gaussians) is applied to these coefficients, original information that it carried is lost.

Performance of different system setups is shown in Table 5.8. Features are $0^{th} - 12^{th}$ PLP coefficients + energy without Δ and $\Delta\Delta$ if not written else.

Configuration	EER
mean + variances, Δ and $\Delta\Delta$	3.66%
DCT - 3 coefficients	4.50%
DCT - 5 coefficients	3.60%
DCT - 10 coefficients	3.77%
DCT - 5 coefficients, Δ and $\Delta\Delta$	3.72%
2 Gaussians in mixture	3.63%
2 Gaussians in mixture, Δ and $\Delta\Delta$	3.80%

Table 5.8: Comparison of equal error rate (EER) of different system configurations.

DCT coefficients are taken from the first one computed, e.g. from the lowest frequencies. Notice the degradation of system performance when Δ and $\Delta\Delta$ are extracted.

After this series of tests, the optimal achieved configuration is following:

- HCopy extracting tool
- $0^{th} - 12^{th}$ PLP coefficient + energy
- Δ and $\Delta\Delta$ of the features
- mean + variance modification method

5.2 SAD system based on SVM

The goal of this phase is to use the best features from previous tests and to build speech activity detection system on them. First, input file is cut into segments of the length l_i with shift s_e. Value of l_i is the same as the one used for model training. These segments are classified and output probabilities are converted to binary form and smoothed. With known l_i and s_e, output file with speech labels are reconstructed and set as an input of evaluation script. Values of FA and MISS are then read out from script output.

5.2.1 Smoothing method

Two ways how to transform output classification probabilities to binary state form are tested. First method uses thresholding and simple rule to remove short segments, second is based on viterbi decoder. Input of thresholding method is vector of probabilities. These are assigned to one of the classes using hard decision (threshold). Resulting vector of binary states is smoothed by rule described in section 3. Input of viterbi based system is also vector of probabilities. To have direct comparison with thresholding method, the same value of

threshold th as used by previous method is subtracted from input. Resulting values are the actual input of viterbi algorithm, its output is vector of binary states. To determine the optimal value of transition probabilities, viterbi smoothing system was run on vector of input probabilities with values of $th =< -0.8, 0 >$ subtracted. Transition probabilities were tested from following interval $sp_s = ns_s =< 0.5, 1 >, sp_c = ns_c = 1 - sp_c$ for each th. Values of FA and MISS rate were averaged for each sp_c and graph in Figure 5.4 was plotted.

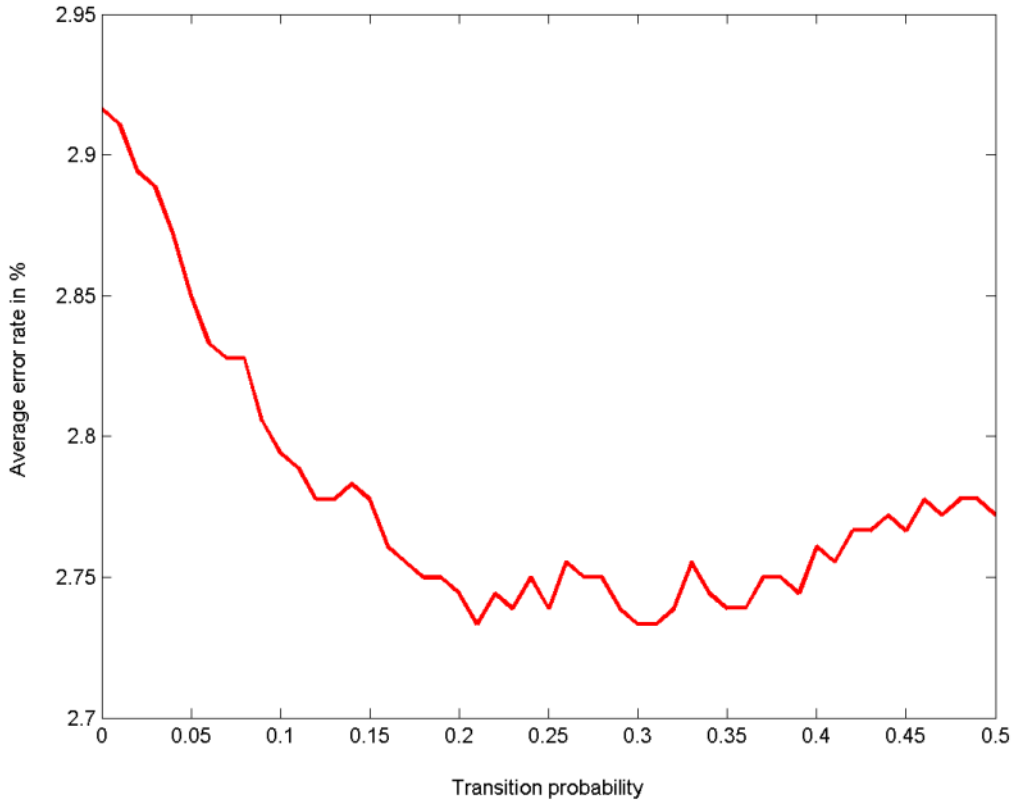


Figure 5.4: Dependency of averaged error rates on transition probability sp_c.

Graph shows that sp_c from interval $< 0.2, 0.3 >$ brings very similar results of average error rates. The middle of this interval is chosen, transition probabilities are set to $sp_s = ns_s = 0.75, sp_c = ns_c = 0.25$. Comparison with rule smoothing method is made with this configuration, variable for DET curve plotting is threshold th from interval $< -1.5, 1.2 >$. DET curves for model with parameters $s_e = 50ms, l_i = 50ms$ are shown in Figure 5.5.

Curves have very similar shape, but rule smoothing method is always a step ahead.

5.2.2 Model and step size

When choosing the best model and step combination, value s_e is taken from interval (0,l_i¿, where l_i is length of segments for given model. When $s_e < l_i$, not only segment itself, but also its close neighborhood is used for its classification. If $s_e = l_i$, just the duration of segment is used for classification. Value of s_e cannot be longer than l_i, because some of the input would remain unclassified. All three possibilities are shown in Figure 5.6.

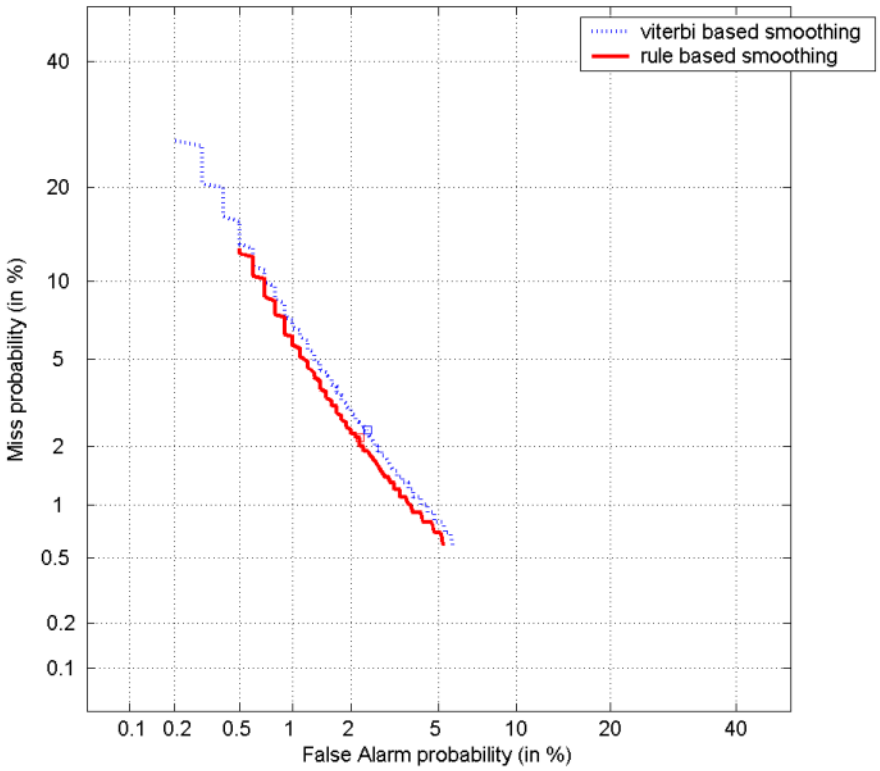


Figure 5.5: DET curves for viterbi and smoothing rule method, using model with parameters $s_e = 50ms, l_i = 50ms$.

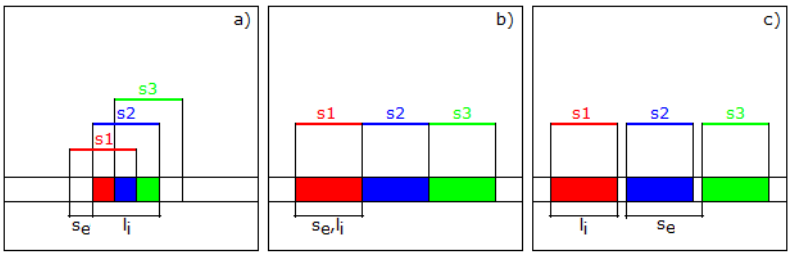


Figure 5.6: a) Length of segments l_i is bigger than shift s_e. Interval s_x is used for the x^{th} segment classifying. b) Length of segments is the same as shift. Only duration of segment itself is used for classifying. c) Shift is bigger than length of the segment. There are unclassified spaces between intervals s_x.

Values of s_e from 50ms to l_i are tested for each model. Step of s_e is 50ms. Graph in Figure 5.7 shows EER of all possible configuration combinations, systems uses rule method smoothing.

Test shows that models working with bigger segment sizes suffers from their lower reso-

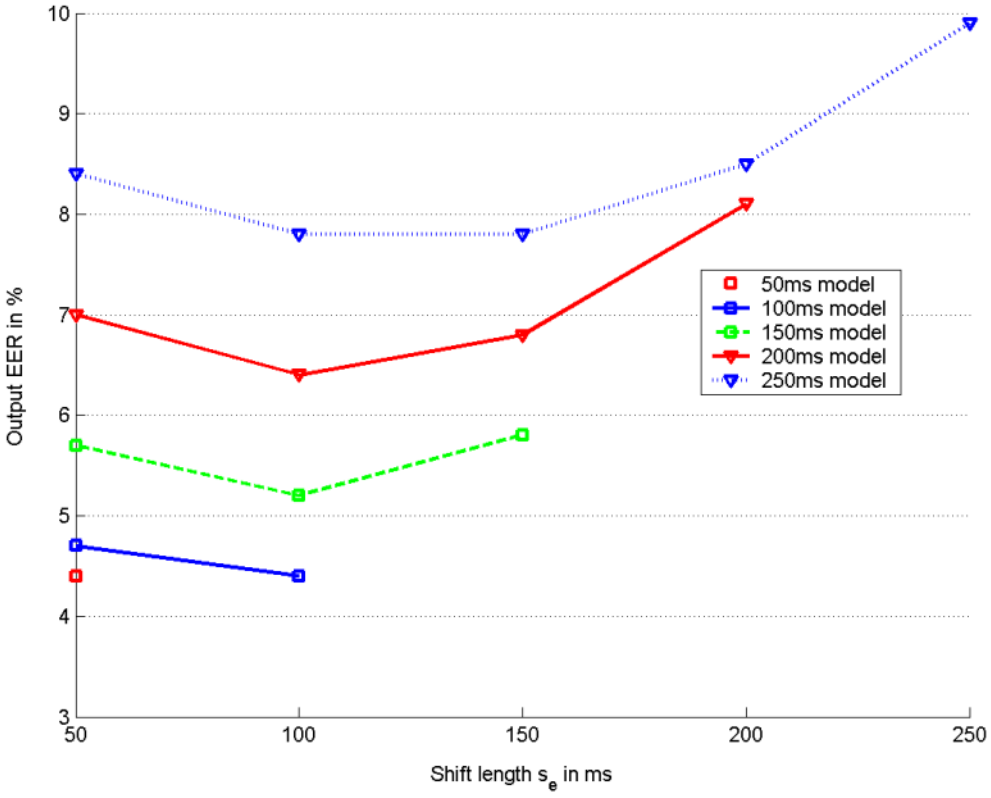


Figure 5.7: EER of all possible model-s_e combinations.

lution, because for example 250ms segment is quite long and the occurrence of speech/non-speech class transition is much more probable than in shorter segments. Segments with class transition have indefinite class belonging probability and are hard to process for SAD system. Noticeable think is also existence of some dependency of l_i on s_e. The bigger length of processed segments l_i, the bigger shift s_e is necessary to achieve optimal performance. The best EER rate is observed at two system configurations: $l_i = 50ms, s_e = 50ms$ and $l_i = 100ms, s_e = 100ms$. DET curve in Figure 5.8 is plotted to further explore behavior of these two configurations.

Both configurations produces very similar results in area around EER point. Model working with shorter segments outperforms the other one when FA rate is low, but is worse on the other side of the curve. This behavior is hard to explain and would need some more tests to understand it. Results for individual files for both top performing models are listed in Table 5.9. Numbers correspond to EER point on DET curve.

Noticeable thing was that model with $l_i = s_e = 50ms$ uses very aggressive threshold $th = -0.5$, which means that lot of non-speech samples is classified as speech and result depends more on smoothing. Model with parameters $l_i = s_e = 100ms$ uses a bit more conservative threshold $th = -0.28$, so more work relies on classification.

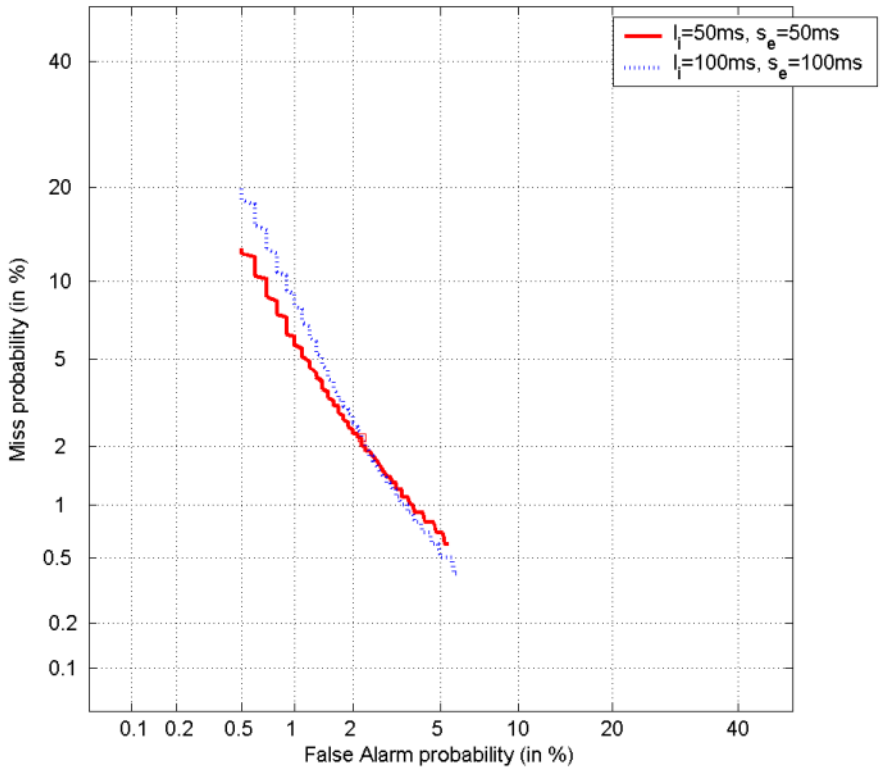


Figure 5.8: DET curves for the two best configurations, red solid line for $l_i = 50ms, s_e = 50ms$ and blue dotted line for $l_i = 100ms, s_e = 100ms$.

	FA_{50ms}	$MISS_{50ms}$	FA_{100ms}	$MISS_{100ms}$
ert3_20061022_195946	1.8%	2.7%	3.0%	2.0%
ert3_20061023_191042	3.6%	1.5%	1.5%	3.3%
ert3_20061023_191624	2.3%	1.8%	1.6%	2.3%
ert3_20061023_195504	1.6%	3.0%	3.2%	1.5%
ert3_20061024_183258	2.4%	1.9%	1.9%	2.2%
ert3_20061102_185334	2.3%	2.0%	2.0%	2.3%
Overall	**2.2%**	**2.2%**	**2.2%**	**2.2%**

Table 5.9: Comparison of values of FA and MISS of the best performing systems for individual files. Numbers correspond to EER point on DET curve.

5.3 SAD system based on phoneme recognizer

Basic idea of this system is that output of phoneme recognizer is list of phonemes, which can be divided into speech and non-speech ones. So if all the speech ones are concerned as speech class, and non-speech as silence class, simple speech activity detection is done. However this system as it is has relatively high miss rate, because there is small gap between each two recognized phonemes. This is solved by smoothing the classification result. If silence duration between two speech intervals is smaller then t_r, this silence interval and

two neighboring speech intervals are joined together and marked as speech.

Used phoneme recognizer phnrec is trained on Hungarian part of SpeechDat-E database [12] and its dictionary contains 58 phonemes and 4 labels for silence event.

System is tested following way. Recognized phonemes are replaced by speech/non-speech labels according to a class of the phoneme. Then smoothing is applied, removing silence gaps between two speech labels up to length t_r, which is taken from interval $< 0, 140 > ms$. Smoothed output is scored by evaluation script and FA and MISS rates are plot into DET curve. Results are shown in Figure 5.9.

System without smoothing has error characteristics listed in Table 5.10.

	EER
FA	0.6%
MISS	15.4%

Table 5.10: Error characteristics of SAD system based on phoneme recognizer without smoothing.

When smoothing is applied, measured EER is 5.65%. This result is for $t_r = 54ms$. Because all the tests are performed on the same dataset which is used to evaluate SVM based system, it is possible to directly compare results of these two systems.

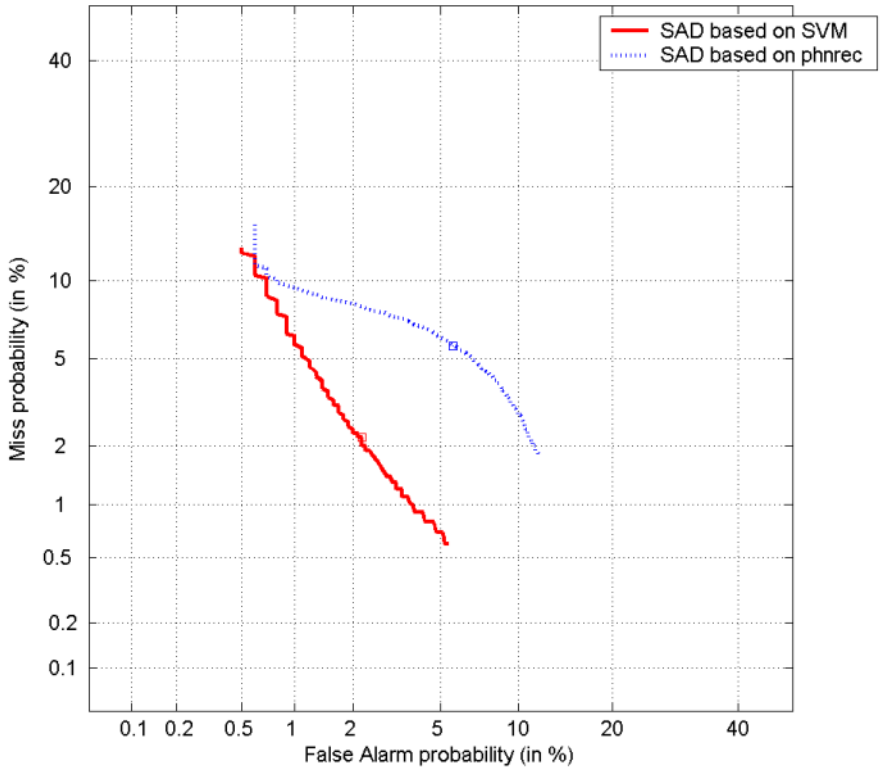


Figure 5.9: Comparison of two DET curves, red solid line is for system based on SVM, blue dotted for system based on phnrec.

Comparison shows that SVM based system outperforms system based on phnrec on whole interval. The only comparable area is at the beginning of the curve, where MISS rate is high. That is because phnrec is very good in detecting isolated phonemes, and this part of the DET curve belongs to non or just a little smoothed phoneme recognizer output. If smoothing is applied more heavily, more silence intervals inside the words are smoothed, but also more intervals between the words is affected, which increases FA rate and decreases system performance. Other reason might be that phnrec is primarily trained and designed to work with meetings data, while here is used on broadcast news data. Broadcast data contain more noise and a lot of other sounds (rumbling crowd behind the TV reporter, car noise), which does not appear in meeting data.

Detailed example of different systems SAD segmentation is made in Figure 5.10. Chosen audio interval is from one file from ERT database and its length is about 17 seconds, from which first 10 is part of the news taken in TV studio, so the noise rate is relatively low, and the second part is outdoor and the level of noise is much higher.

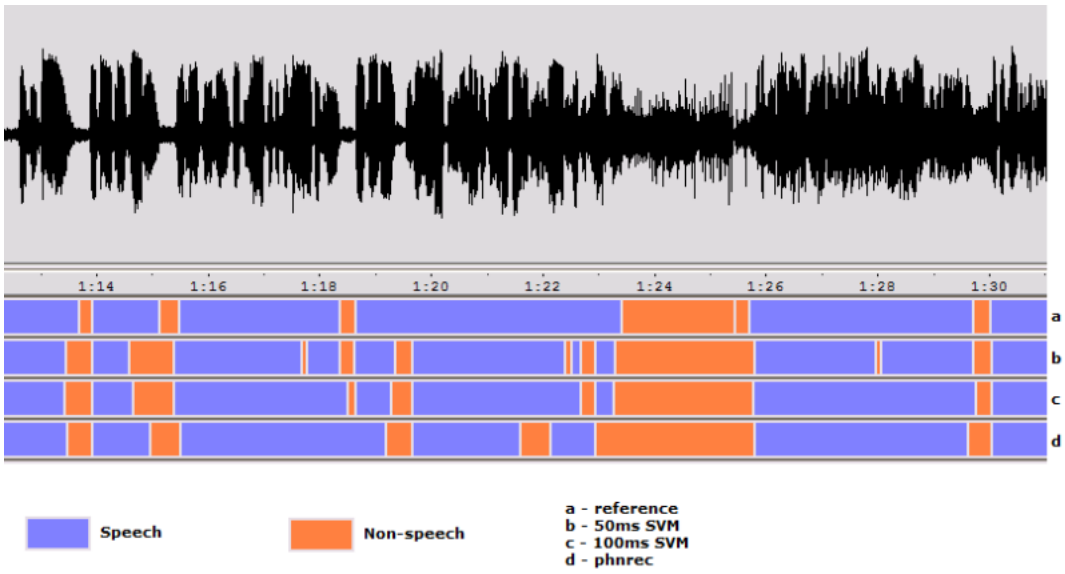


Figure 5.10: Example of tested systems segmentation on a part of one audio file.

There are few things in Figure 5.10, which prove hypothesis from previous sections. Comparison of two SVM models shows that model working with shorter segments has due to this size advantage in better aligned segment boundaries, but as a drawback few short non-speech intervals appeared in the speech parts. However differences are minimal, which results in very similar performance overall. System based on phnrec has bigger errors in segment boundaries and also skips one silence interval, which was smoothed away. Noticeable thing is that all three tested systems detected silence interval between 1:19 and 1:20, but there is no such in reference file. When exploring this part of the audio, it turned out that it is the part where two word are connected by vowel and are pronounced together. Energy is very low in this place, and that is the most probable reason that all systems recognized this area as non-speech.

5.4 Comparison with system based on GMM

Developed system is compared with another SAD system based on different principle. Comparison is made with SAD subsystem of SHoUT [7]. SHoUT is a diarization system based on Gaussian Mixture Models (GMM), but it has also SAD part, as detection of speech activity is preprocessing for diarization algorithm. System itself is not available, but results and development database are. Database contains 27 RT06s conference meetings, from which 19 are used for training and 8 for testing. Evaluation script is the same as the one used for SVM system development, so there were no bigger problems with data migration. The only difference between previously used evaluations is that no-score „collar“ parameter of evaluation script is set not to 0s, but to 0.25s. Help file of used evaluation script defines collar parameter as following: *„No-score collar is the no-score zone around reference speaker segment boundaries. (Speaker Diarization output is not evaluated within +/- collar seconds of a reference speaker segment boundary.)“*. Practically it lowers resulting error rates of the evaluated system. Comparison of DET curves with collar and without collar is in Figure 5.11. SVM system with parameters $s_e = l_e = 50ms$ is used and it is tested on RT06s conference meetings database.

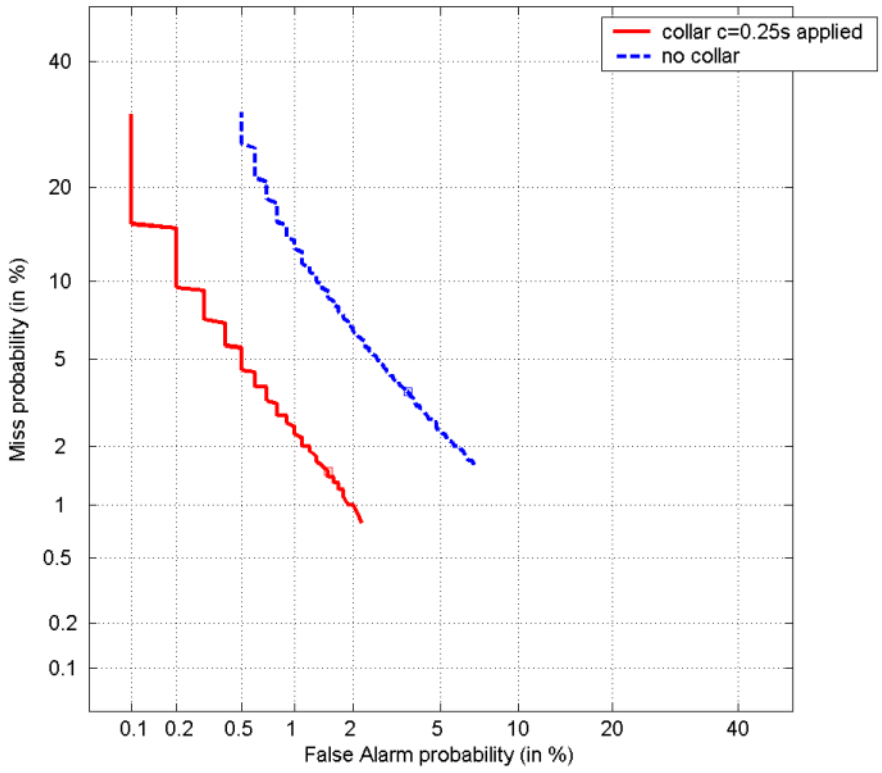


Figure 5.11: Comparison of two DET curves, red solid line is for system with collar set to 0.25s, blue dotted for system without collar.

Value of EER improves significantly, from $EER = 3.6\%$ for system without collar to $EER = 1.5\%$ for system using collar $c = 0.25s$. This value of collar is standardly used in NIST Speaker Diarization Evaluations and its purpose is to remove influence of inaccuracy

of data labeling from system result.

SHoUT system results are listed in form of minimal DER instead of EER used in all previous tests, so minimal value of DER is readout from DET curves to compare the systems. It is defined as

$$s = min(DER) = min(MISS + FA)$$

where s is system error. Values of minimal DER for all compared systems are listed in Table 5.11.

	FA	MISS	DER
SHoUT (GMM)	0.9%	2.1%	3.0%
phnrec (phoneme recognizer)	1.5%	3.5%	5.0%
SVM 50ms	1.0%	1.9%	2.9%
SVM 100ms	0.7%	1.4%	2.1%

Table 5.11: DERs for compared systems on RT06s conference meetings.

Results from the comparison are very good and shows that SVM system working with 50ms segments is comparable with SHoUT and SVM with 100ms even outperforms it. Error rates of system based on phnrec are the worst, but this is in line with its results from previous tests.

Chapter 6

Conclusion

Speech activity detection system is presented in this thesis. Acoustic features are extracted from audio recordings, they are split into segments from which feature vectors are computed. Speech activity detection is then performed by classification of these vectors to speech or non-speech class with a smoothing process.

In the first part of the tests, the best combination of features, feature extracting tools and modification methods is found. The lowest error rates are achieved by 13 MFCC coefficients enhanced by energy plus their delta and acceleration coefficients, extracted by HCopy from HTK toolkit. Means and variances from n consecutive feature vectors are taken and they are concatenated creating SVM training vectors.

The second part of the tests uses results from the first part and is about finding optimal SAD system parameters and smoothing method. Two setups produces equal results, it is system working with 50ms segments and 100ms segments. Simple smoothing method is used to remove small segments from system output. It is based on changing segment class if it is surrounded by segments from different class from both sides.

The last part of this work refers to comparison of developed system with two other systems, based on different principles. The idea of the first approach is to use phoneme recognizer for SAD, where all phonemes are linked to speech class. Second approach uses Gaussian Mixture Models (GMM) to model distribution of features for speech and silence. Tests are made on RT06s conference meeting data and SVM based system produces very competitive results, with the best achieved detection error rate of 2.1%, which is 30% relative improvement over conventional GMM based system.

Future work on this system should concentrate on more detailed testing of system parameters. When tuning system parameters, step of 50ms for segment length and segment overlap was used. It is quite big step compared to segment length and segment overlap, which are both from interval $< 50ms, 250ms >$, but more detailed testing would have been too time and computational sources consuming. Another area of interest might be final speech segments border alignment, but this task is disputable, since most evaluation scripts uses some no-score margin around reference labels, where no error time is added.

Bibliography

[1] P. H. Chen, C. J. Lin, and B. Schölkopf. A tutorial on v-support vector machines. Appl. Stoch. Models. Bus. Ind., 2005.

[2] Non-linear svm.
[online], http://math.u-bourgogne.fr/monge/bibliotheque/ebooks/csa/htmlbook/node221.html. [rev. 2009-05-05], [cit. 2009-05-05].

[3] E. Gouws, K. Wolvaardt, N. Kleynhans, and E. Barnard. Appropriate baseline values for hmm-based speech recognition. In *Proceedings of the 15th Annual Symposium of the Pattern Recognition Association of South Africa*, page 169, Grabouw,South Africa, November 2004.

[4] Hynek Heřmanský. Perceptual linear prediction analysis of speech. *Acoustical Society of America*, 1990.

[5] Htk book. [online], http://users.ece.gatech.edu/ antonio/htkbook/htkbook_tf.html. [rev. 2009-05-05], [cit. 2009-05-05].

[6] Htk toolkit. [online], http://htk.eng.cam.ac.uk/. [rev. 2009-05-05], [cit. 2009-05-05].

[7] M. Huijbregts. *Segmentation, Diarization and Speech Transcription: Surprise Data Unraveled*, volume 08-123 of *CTIT Ph.D. thesis Series*. Centre for Telematics and Information Technology (CTIT), 2008. ISBN 978-90-365-2712-5, ISSN 1381-36-17.

[8] Lie Lu, Hong-Jiang Zhang, and Stan Z. Li. Content-based audio classification and segmentation by using support vector machines. In *Multimedia Systems*. Springer-Verlag, 2003. DOI 10.1007/s00530-002-0065-0.

[9] Markaki M., Karpov A., Apostolopoulos E.and Astrinaki M., Stylianou Y., and Ronzhin A. A hybrid system for audio segmentation and speech-endpoint detection of broadcast news. In *Proceedings of SPECOM*, Moscow,Russia, October 2007.

[10] NIST. Rich transcription 2006 spring meeting recognition evaluation plan v2. Rich Transcription 2006 Spring Meeting Recognition workshop, 2006.

[11] Detware.
[online], http://www.nist.gov/speech/tests/spk/2004/. [rev. 2009-05-05], [cit. 2009-05-05].

[12] P. Pollak, J. Cernocky, J. Boudy, K. Choukri, H. van den Heuvel, K . Vicsi, A. Virag, R. Siemund, et al. Speechdat(e) - eastern european telephone speech databases. In

Proceedings LREC'2000 Satellite workshop XLDB - Very large Telephone Speech Databases, pages 20–25, Athens,Greece, May 2000.

[13] Sprachcore speech recognition software.
[online], http://www.icsi.berkeley.edu/~dpwe/projects/sprach/sprachcore.html. [rev. 2009-05-05], [cit. 2009-05-05].

[14] Support vector machines.
[online], http://en.wikipedia.org/wiki/Support_vector_machine. [rev. 2009-05-05], [cit. 2009-05-05].