Institut für
Technische Informatik und
Kommunikationsnetze

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Text-Independent Speaker Verification with HMM-Based Abstract Acoustic Elements

Semester Thesis

TIK Institute, D-ITET, ETH Zurich

Author
Thomas Willi

Supervision
Dr. Beat Pfister, Tofigh Naghibi

January 17, 2013

Contents

Chapter 1

Introduction

1.1 Abstract

In this semester thesis, a new approach for text-independent speaker verification with so called abstract acoustic elements (AAEs) has been investigated. Basically, AAEs are partitions of the acoustic space, where each AAE has its own probability distribution describing its set of training features. The idea is then to assign speech samples to the most likely sequence of AAEs and to describe the speech of different speakers using these AAEs or rather the (speaker-specific) transition probabilities between them. The AAEs have been computed using the LBG algorithm.

For speaker verification, we need speaker-specific models on the one hand and a universal background model (UBM) on the other hand. Baum-Welch and Viterbi training have been used to train such models based on zero and first order hidden Markov models (HMMs).

Furthermore, the Viterbi and the forward algorithm have then been used to compute the likelihoods of test speech samples for the stochastic models. These likelihoods are required to compute likelihood ratios, which are the basis of decision-making for speaker verification.

The main result of the experiments is that increasing the number of AAEs is promising with respect to improving the performance, i.e. decreasing the probabilities of false rejection and false acceptance. Unfortunately, a further increase (more than 32 AAEs) would be computationally very intensive and more speaker-specific training data would be required.
The approach of increasing the influence of the speaker-specific transition probabilities on the decision step of the speaker verification did not improve the performance. Unfortunately, the results were even worse.

For a schematic representation of the whole project, please have a look at Figure 1.1.

1.2 Motivation

The use of abstract acoustic elements is motivated by the fact that they are not changed anymore after their generation. As a consequence, less parameters have to be trained, which means that less training data for the training of speaker-specific models is required. Since the amount of available speaker-specific speech samples is often very limited, this may be beneficial.

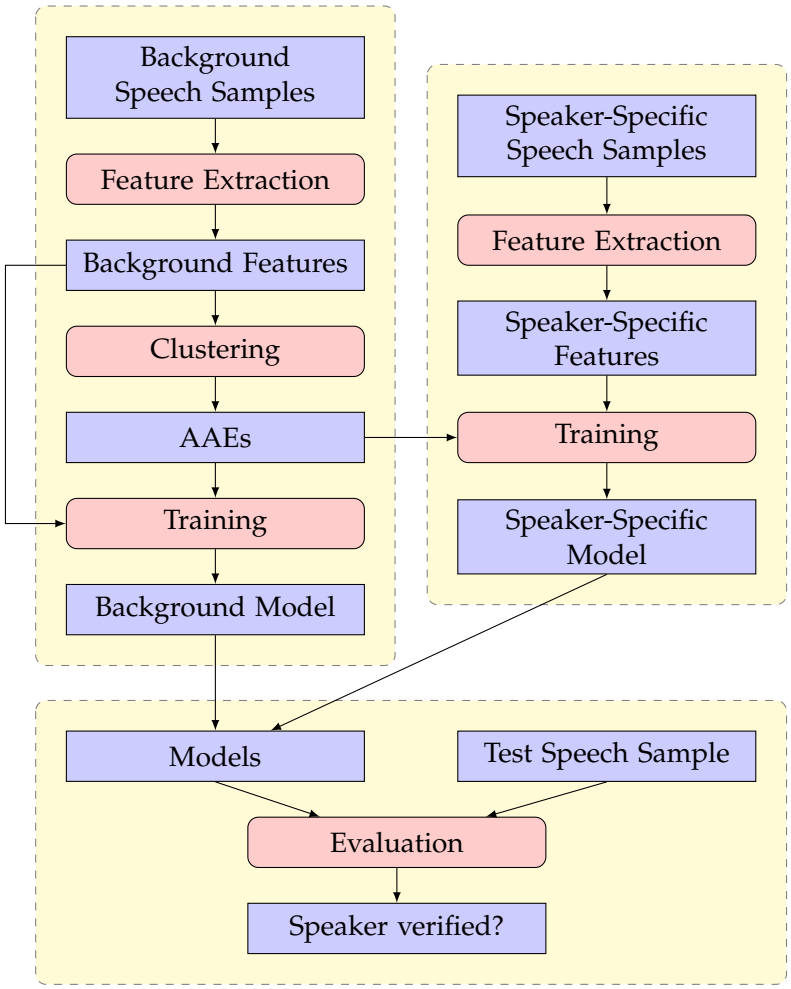

Figure 1.1: Project overview (top left part: generation of abstract acoustic elements and a universal background model; top right part: generation of a speaker-specific model; bottom part: actual speaker verification)

Chapter 2

Generation of Stochastic Models

2.1 Feature Extraction

To train stochastic models, training data in form of feature vectors is required. These feature vectors are computed by analyzing speech samples.

In this thesis, standard features, namely *mel frequency cepstral coefficients (MFCCs)*, have been used. As in the laboratory exercises, the number of MFCCs per feature vector has been set to twelve.

For more information on MFCCs, see [1], Section 4.6.5.

In the implementation, the Matlab function `mfcc.m` from the laboratory exercises has been used for the feature extraction. The latter evaluates the MFCCs according to the ETSI standard.

2.2 Clustering & Abstract Acoustic Elements

Succeeding the feature extraction, the LBG algorithm (standard clustering algorithm; cf. [1], Section 4.7.2.2) has been used to divide the feature space into P partitions. These partitions represent the so called *abstract acoustic elements (AAEs)*. Figure 2.1 shows an example of the clustering for 2-dimensional synthetic feature vectors.

Since the approach of this thesis is to describe each AAE by a single Gaussian mixture component (see [1], Appendix A.4.2.2 for more information), each AAE can be described by its mean vector (mean of its training feature vectors) and its covariance matrix, which describes the correlation between the components of the feature vectors.

In this thesis, the components of the feature vectors have been assumed to be uncorrelated. With this assumption, the covariance matrix reduces to a diagonal matrix, which can be stored as a single vector. The components of such a vector are the variances of each dimension of the feature space.

Since the LBG algorithm directly delivers the mean vectors for all partitions, we just need to compute the variance vectors in order to get all the data required to describe the AAEs. The variances can easily be computed using the following maximum likelihood estimator for Gaussian distributions:

$$\tilde{\sigma}_i^2 = \frac{1}{K_n} \sum_{k=1}^{K_n} (x_{k,i} - \tilde{\mu}_{n,i})^2,$$

where K_n is the number of training feature vectors of the n-th AAE and i is the dimension.

Remark:

Note that the LBG algorithm may end up in a local minimum of the mean distortion instead of a global one.

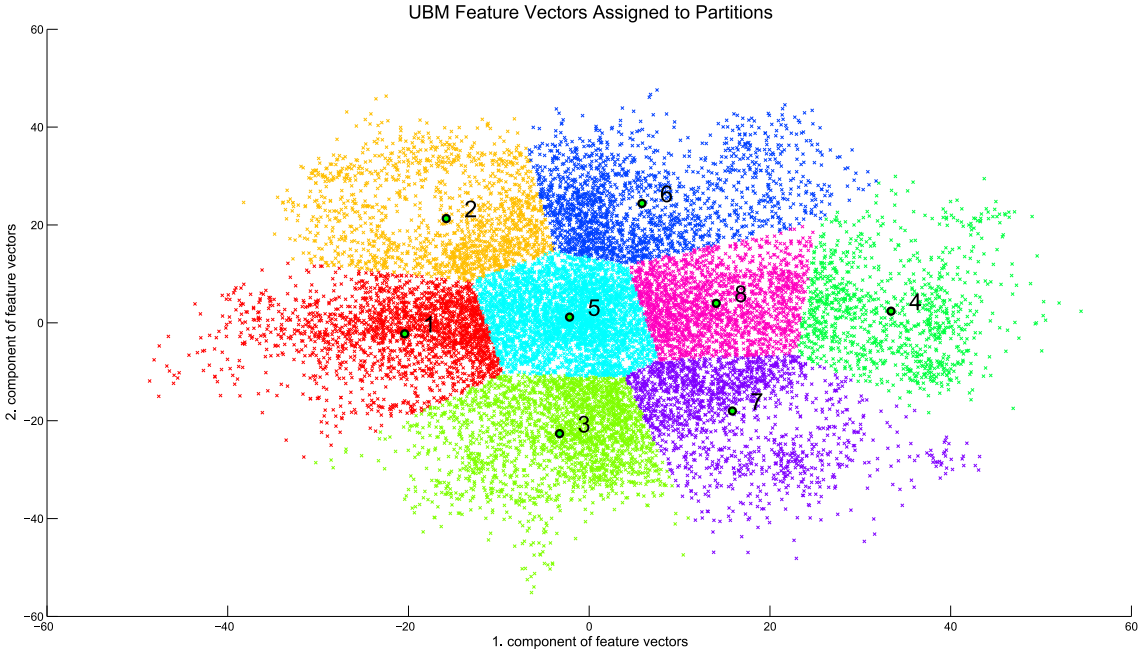

Figure 2.1: Example for clustering: random 2-dimensional training feature vectors assigned to partitions, where the green dots represent the mean vector of each partition.

2.3 Structure of the Stochastic Models

The idea of the approach in this thesis is to characterize a specific speaker by the frequency of occurrence of the AAEs or rather the frequency of the transitions from one AAE to another in his or her speech. Therefore, we need to assign speech samples to the most likely sequence of AAEs. A feature vector can be assigned to the most likely AAE by evaluating the Gaussian distribution for each AAE and finding the one with the highest likelihood. But to find the most likely sequence of AAEs, we also need to take into account the probabilities of the AAEs following each other. For this reason, we introduce a characteristic set of transition probabilities for each model.

Let us define the set of all emitting states as $\mathcal{A} \triangleq \{A_2, \ldots, A_{N-1}\}$, where $N = P + 2$. These emitting states represent the AAEs. Additionally, we define an initial state A_1 and a final state A_N. Each speech sample of length T (number of feature vectors) can then be described by a sequence of AAEs. Let us denote this sequence by $\mathbf{Q} \triangleq Q_1, \ldots, Q_T$, where Q_1 and Q_T are always A_1 and A_N, respectively.

Have a look at Figures 2.2 and 2.3, which should give a rough idea of how the distribution of the speaker-specific training feature vectors could lead to different transition probabilities although randomly generated synthetic data has been used for these plots. Note that the partitions or rather the mean vectors are always those computed with the UBM feature vectors in Figure 2.1.

2.3.1 Universal Background Model and Speaker-Specific Models

For speaker verification, we need speaker-specific models on the one hand, but also a *universal background model (UBM)* on the other hand. The reason for that is that we want to test whether the likelihood of a specific speech sample is significantly higher for a speaker-specific model than for the UBM, which should represent the average of all possible speakers.

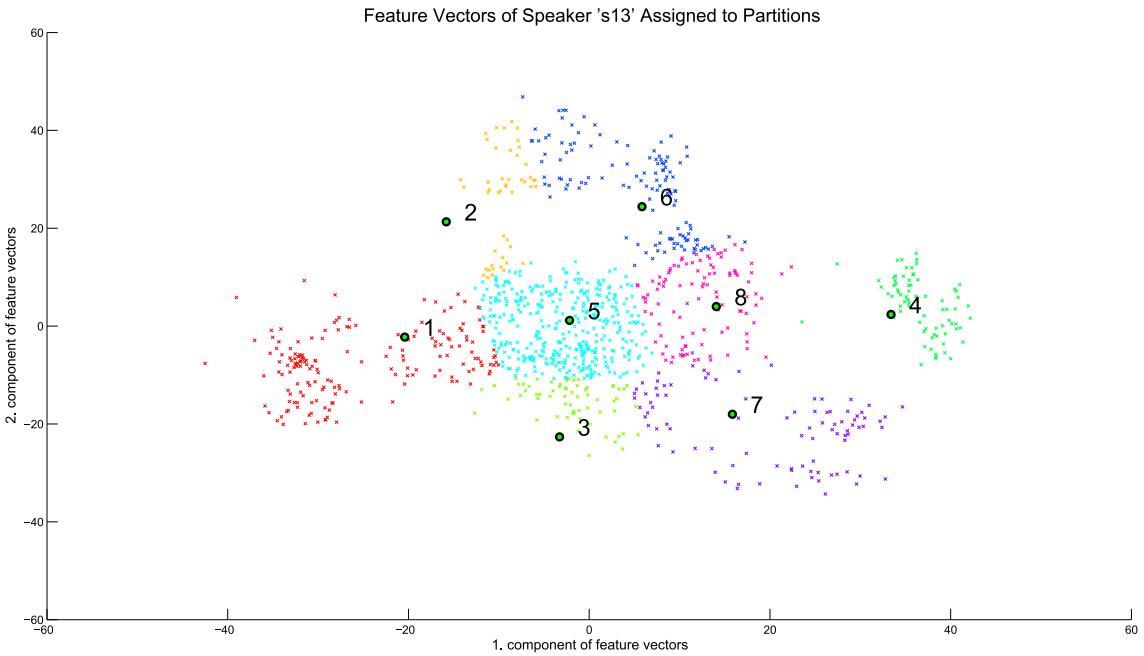

Figure 2.2: Distribution of synthetic training features for speaker ’s13’

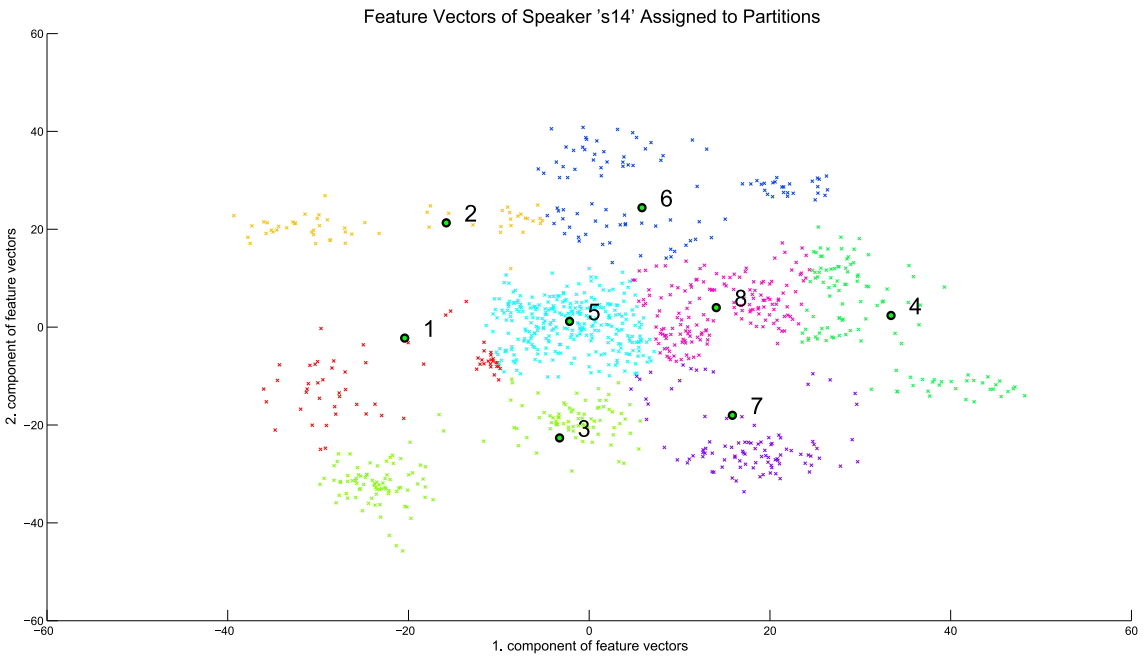

Figure 2.3: Distribution of synthetic training features for speaker ’s14’

The UBM and the speaker-specific models for this approach basically consist of the same parts, namely a set of transition probabilities and a *Gaussian mixture model (GMM)*. Speaker-specific is only the set of transition probabilities, whereas the GMM is the same for all models since it represents the AAEs.

2.3.2 Transition Probabilities

Let us first have a look the the set of transition probabilities. In this thesis, two different types of transition probabilities have been investigated:

- unigram transition probabilities
- bigram transition probabilities

Unigram Models

The simplest approach for transition probabilities is a set of unigram transition probabilities. If we assume unigram transition probabilities, we assume that Q_t does not depend on any previous state in $\mathbf{Q}$, i.e.

$$P(Q_t|Q_1,\ldots,Q_{t-1}) \approx P(Q_t), \text{ where } Q_i \in \mathcal{A} \cup \{A_1, A_N\}$$

Figure 2.4 visualizes this characteristic by the fact that every emitting state $A_2,\ldots,A_{N-1}$ is followed by the non-emitting state A_N, wherefrom we return to the also non-emitting initial state A_1 with probability one.

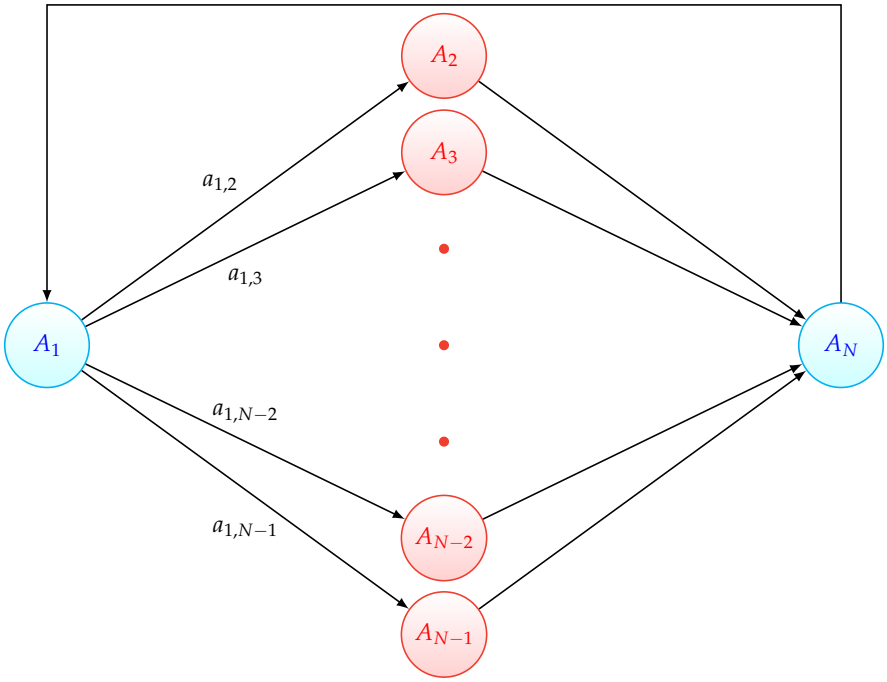

Figure 2.4: Unigram network (blue: non-emitting state; red: emitting state)

Since the probability of a transition from A_1 to A_N is equal to one, A_N can be neglected, i.e. the probability of a transition from A_i to A_1 is equal to one, where $i \in \{2,\ldots,N-1\}$.

With this insight, we get a transition matrix of the following form:

$$A_{\text{unigram}} = \begin{bmatrix} 0 & a_{1,2} & a_{1,3} & \cdots & a_{1,N-1} & 0 \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \qquad (N \times N),$$

where $a_{1,j}$ is the probability of appearance of state A_j. Note that $a_{1,N}$ is zero to exclude empty observation sequences.

For more information on unigram models, see [1], Section 13.2.4.1.

Bigram Models

Another approach for transition probabilities is a set of bigram transition probabilities, where we assume that Q_t only depends on previous state Q_{t-1} in $\mathbf{Q}$, i.e.

$$P(Q_t|Q_1, \ldots, Q_{t-1}) \approx P(Q_t|Q_{t-1}), \text{ where } Q_i \in \mathcal{A} \cup \{A_1, A_N\}$$

A general representation of a set of bigram transition probabilities can be seen in Figure 2.5.

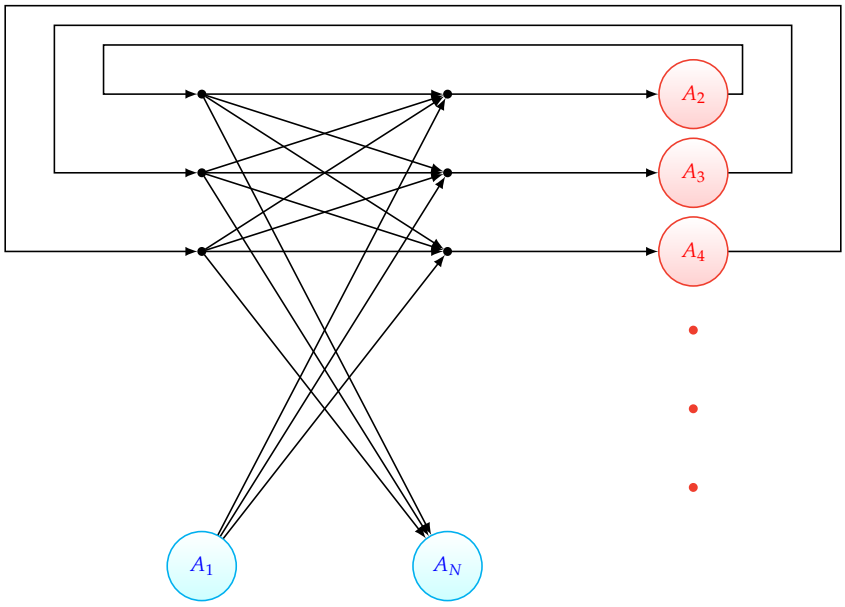

Figure 2.5: Bigram network (blue: non-emitting state; red: emitting state)

For bigram transition probabilities, we get a transition matrix of the following form:

$$A_{\text{bigram}} = \begin{bmatrix} 0 & a_{1,2} & a_{1,3} & \cdots & a_{1,N-1} & 0 \\ 0 & a_{2,2} & a_{2,3} & \cdots & a_{2,N-1} & a_{2,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{N-1,2} & a_{N-1,3} & \cdots & a_{N-1,N-1} & a_{N-1,N} \\ 0 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \qquad (N \times N)$$

where $a_{i,j}$ is the probability of a transition to state A_j when being in state A_i. Note that $a_{1,N}$ is zero to exclude empty observation sequences.

For more information on bigram models, see [1], Section 13.2.4.2.

2.3.3 Gaussian Mixture Model

For each AAE A_j, a Gaussian mixture model has been used to describe the distribution of its training feature vectors. A GMM is of the following form:

$$b_j(\mathbf{x}) \;=\; \sum_{k=1}^{M} c_{jk}\, b_{jk}(\mathbf{x}) \;=\; \sum_{k=1}^{M} c_{jk}\, \mathcal{N}(\mathbf{x}, \boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}) \;,$$

where $b_j(\mathbf{x})$ is the observation likelihood of the feature vector $\mathbf{x}$ and c_{jk} is the weighting coefficient of the k-th mixture component.

In this thesis, only GMMs with a single mixture component ($M = 1$, $c_{j1} = 1$) have been used. Note that this is not really a restriction, since we could just increase the number of AAEs.

For more information on GMMs, cf. [1], Appendix A.4.2.2.

2.4 Training

The transition probabilities of the stochastic models have been trained using Baum-Welch (cf. [1], Sections 5.4.7 and 5.5.4) or Viterbi (cf. [1], Sections 5.4.8 and 5.5.5) training.

2.4.1 Required Amount of Training Data

For the training of stochastic models, it is important to have enough training data. Otherwise, a trained model may be a bad representation of the actual characteristics.

As a rule of thumb, at least ten feature vectors for each parameter to train are required assuming a uniform distribution of the feature vectors. If the distribution is not uniform, even more training samples are required.

In our case, we get the following expressions for the minimum number of feature vectors F_{min}:

Unigram Models

$$F_{\text{min,UBM}} \;=\; \underbrace{(N-2)\cdot(D+D)}_{\text{AAEs}} \;+\; \underbrace{N-2}_{\text{transition probabilities}}$$

$$F_{\text{min,speaker}} \;=\; \underbrace{N-2}_{\text{transition probabilities}}$$

Bigram Models

$$F_{\text{min,UBM}} \;=\; \underbrace{(N-2)\cdot(D+D)}_{\text{AAEs}} \;+\; \underbrace{(N-2)^2}_{\text{transition probabilities}}$$

$$F_{\text{min,speaker}} \;=\; \underbrace{(N-2)^2}_{\text{transition probabilities}}$$

2.5 Implementation

The whole implementation is based on code of the laboratory exercises 10, 19, 20, 21 and 22 of the speech processing course.

The computation of the probabilities happens in the logarithmic domain.

For more information on the used functions and scripts, please have a look at Appendix B.

Chapter 3

Experiments

3.1 Data Used for Training and Testing

The following three different types of data have been used for training and testing:

- Synthetic data
- Speech samples from exercise 22 of the speech processing course
- TIMIT data base

Synthetic Data

For test purposes, a Matlab script which randomly generates D-dimensional synthetic feature vectors, has been implemented. These feature vectors follow Gaussian distributions around C different mean vectors for a given number of speakers. In other words, this approach leads to a set of C (overlapping) clusters of feature vectors for each speaker. A specific speaker is then described by the distribution of the clusters (mean and variance for each dimension). Note that C has not to be equal to the number of AAEs used later.

Since the synthetic features have only been used for test purposes an delivered no relevant results, no results for synthetic data are included in this report.

Speech Samples from Exercise 22

The used speech data from exercise 22 consist of 825 speech samples from 15 different speakers, where the speech samples are digits repeated five times each.

A possible problem for this set of speech data is that the available amount of training data per speaker is very limited (around 2500 feature vectors). Also the short length of the speech samples may be a problem for the testing because a single speech sample only contains around 50 feature vectors, which are really few feature vectors to express the characteristics of a specific speaker. In particular, the distribution of the feature vectors heavily depends on the number the speaker is saying in a specific speech sample as can be seen in Figures 3.1 and 3.2.

For those reasons, the results from experiments with speech data from exercise 22 are not really meaningful and are therefore not included in this report.

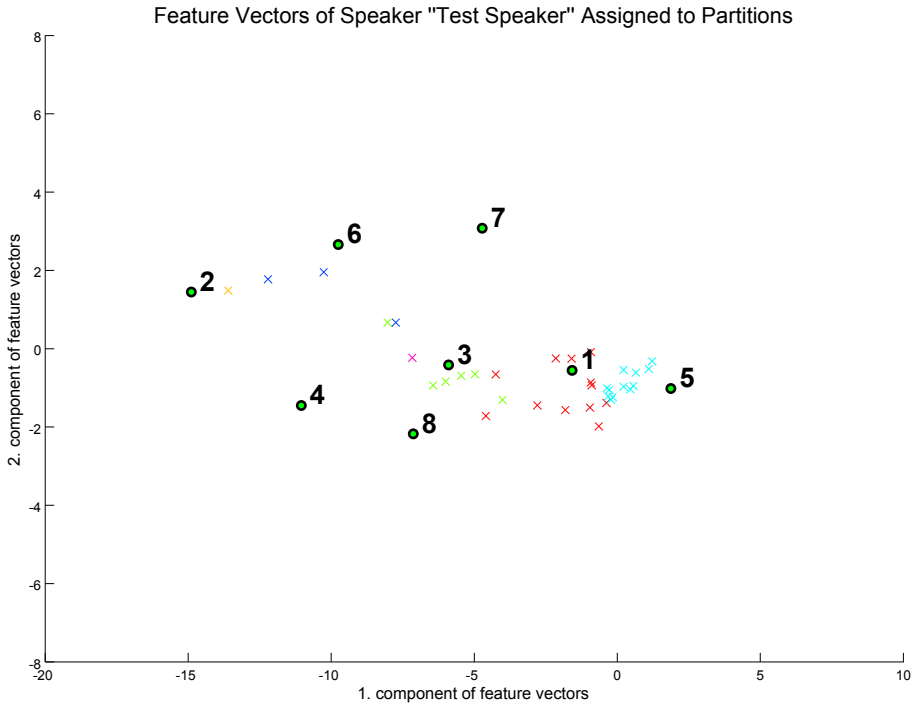

Figure 3.1: First vs. second component of the feature vectors of `z03_r4_s13_test.wav`

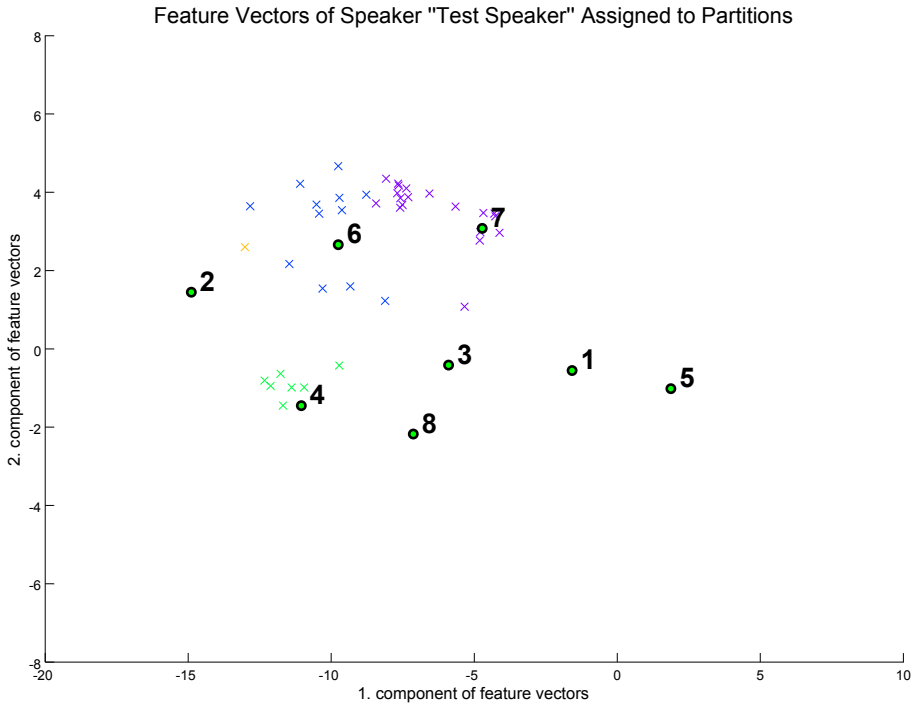

Figure 3.2: First vs. second component of the feature vectors of `z11_r1_s13_test.wav`

TIMIT Data Base

The TIMIT data base is a collection of speech samples of 630 speakers (192 female and 438 male speakers). Unfortunately, the amount of available speech samples per speaker is rather small. There are only eight speech samples per speaker, which means that if one of them is reserved for testing, only seven remain for the training (around 3000 feature vectors).

3.2 Evaluation Algorithms

In this thesis, two different evaluation algorithms have been used, namely the *Viterbi algorithm* (cf. [1], Sections 5.4.4 and 5.5.2) and the *forward algorithm* (cf. [1], Sections 5.4.2 and 5.5.1).

The difference between the outputs of the two algorithms is, that the Viterbi algorithm returns

$$P(\mathbf{X}, \hat{\mathbf{Q}}|\lambda)$$

whereas the forward algorithm returns

$$P(\mathbf{X}|\lambda)\,,$$

where:

- $\mathbf{X} = \mathbf{x}_1, \ldots, \mathbf{x}_T$: Sequence of test feature vectors
- $\hat{\mathbf{Q}} = \hat{Q}_0, \ldots, \hat{Q}_{T+1}$: Most likely sequence of AAEs
- λ: Model (transition probabilities and GMM)

3.3 Parameters & Settings

- P: Number of AAEs: Has to be a power of two due to the use of the LBG algorithm.
- N Total number of emitting (AAEs) and non-emitting states ($N = P + 2$)
- D: Dimension of feature vectors (number of mel frequency cepstral coefficients)
- I: Number of training iterations for Viterbi or Baum-Welch training
- c_A: Weighting factor to increase the influence of the transition probabilities relative to the influence of the observation probabilities, i.e. the logarithms of the transition probabilities are multiplied by c_A, which is equivalent to the transition probabilities to the power of c_A in the linear domain.
  The weighting is only possible for Viterbi evaluation, where the contribution of the transition probabilities to the likelihood of the observation sequence along the optimal path is scaled after the the optimal path has been found.
- κ: Factor to scale the variances of the partitions representing the AAEs, i.e. the variances of the AAEs are multiplied by κ for the evaluation.
- η: Threshold for speaker verification

3.3.1 TIMIT Settings

The software has been designed in such a way that it is very easy to select the desired speakers of the TIMIT data base. The TIMIT data base has been alphabetically sorted, where the name of each speaker consists of a code of four letters and an additional letter ('f' or 'm') in front of them indicating the gender. The desired speakers can then be selected by logical indexing, i.e. to select a specific speaker, its logical index needs to be set to one.

Overview of Settings

Here is an overview of the settings used in the experiments in Section 3.4.1:

- UBM IDs women: Indices $\in \{1,\ldots,192\}$ of alphabetically sorted female speakers selected for the training of the UBM
- UBM IDs men: Indices $\in \{1,\ldots,438\}$ of alphabetically sorted male speakers selected for the training of the UBM
- reg. women: Indices $\in \{1,\ldots,192\}$ of alphabetically sorted female speakers selected for speaker-specific training
- reg. men: Indices $\in \{1,\ldots,438\}$ of alphabetically sorted male speakers selected for speaker-specific training
- test file ID: Index $\in \{1,\ldots,8\}$ of alphabetically sorted speaker-specific speech samples selected for testing

3.4 Speaker Verification

Doing speaker verification, it is possible that a speaker is successfully verified although the corresponding likelihood ratio is not the highest one, i.e. there is the possibility that the likelihood ratio of another registered speaker is much higher.

For this reason, the performance of the speaker verification - which is in a fact a classical decision problem - will be described by the probability of false rejection and the the probability of false acceptance in the following.

Probability of False Rejection

The probability of false rejection, is the probability that the likelihood ratio of a specific speaker is smaller than a given threshold η although it should actually be greater. In other words, it is the probability that the speaker verification fails although the tested speaker really is the claimed one.

Probability of False Acceptance

The probability of false acceptance, is the probability that the likelihood ratio of a specific speaker is greater than a given threshold η although it should actually be smaller. In other words, it is the probability that the speaker verification is successful although the tested speaker is not the claimed one.

3.4.1 Experiments with TIMIT Data Base

Experiment 1

For this experiment, the following settings have been chosen:

Figure 3.3: Settings for experiment 1

P	D	I	c_A	κ	UBM IDs women	UBM IDs men	reg. women	reg. men	test file ID
8	12	5	1	1	1:20	1:20	51:100	51:100	1

With these settings, we get the relation between the probabilities of false acceptance and false rejection shown in Figures 3.4 and 3.5. As we can see, there is no big difference between the evaluation with the Viterbi algorithm (Figure 3.4) and the evaluation with the forward algorithm (Figure 3.5). Therefore, only the results for the Viterbi evaluation will be shown in the following.

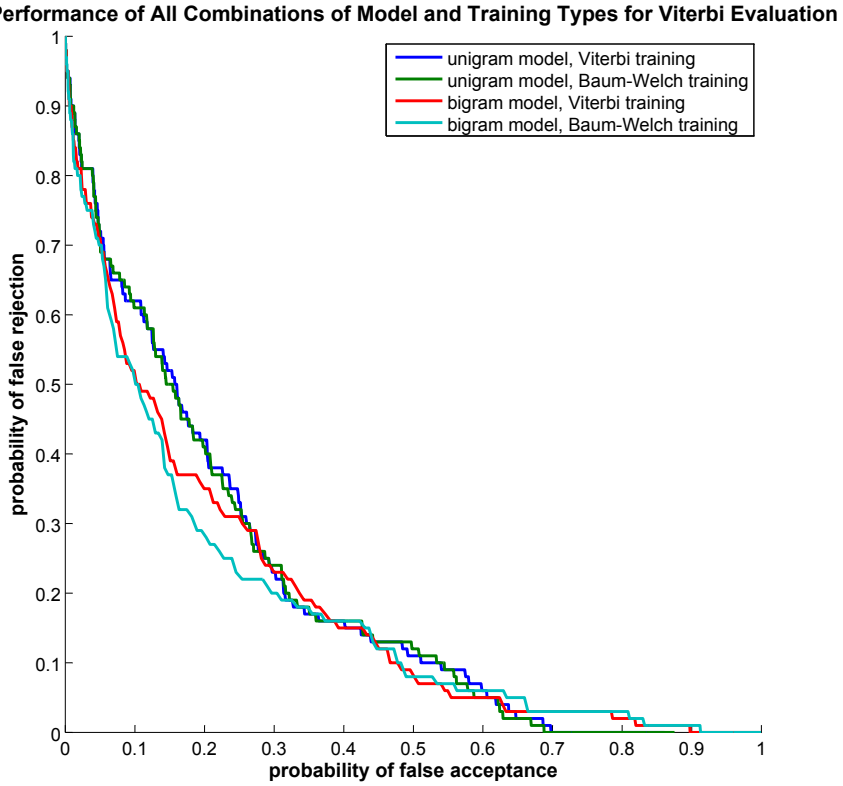

Figure 3.4: Relation between probability of false rejection and probability of false acceptance with Viterbi evaluation for settings of Figure 3.3

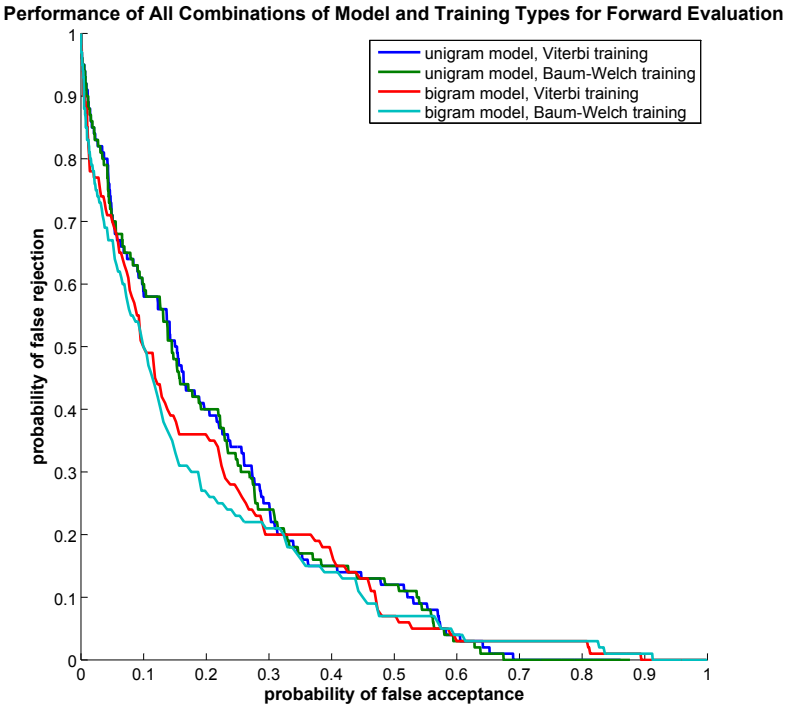

Figure 3.5: Relation between probability of false rejection and probability of false acceptance with forward evaluation for settings of Figure 3.3

An interesting insight is given by Figure 3.6. It shows that the ratio of the probability of false rejection and the probability of false acceptance is much higher for bigram than for unigram transition probabilities for a given threshold η. The plot for the forward evaluation shows the same characteristics.

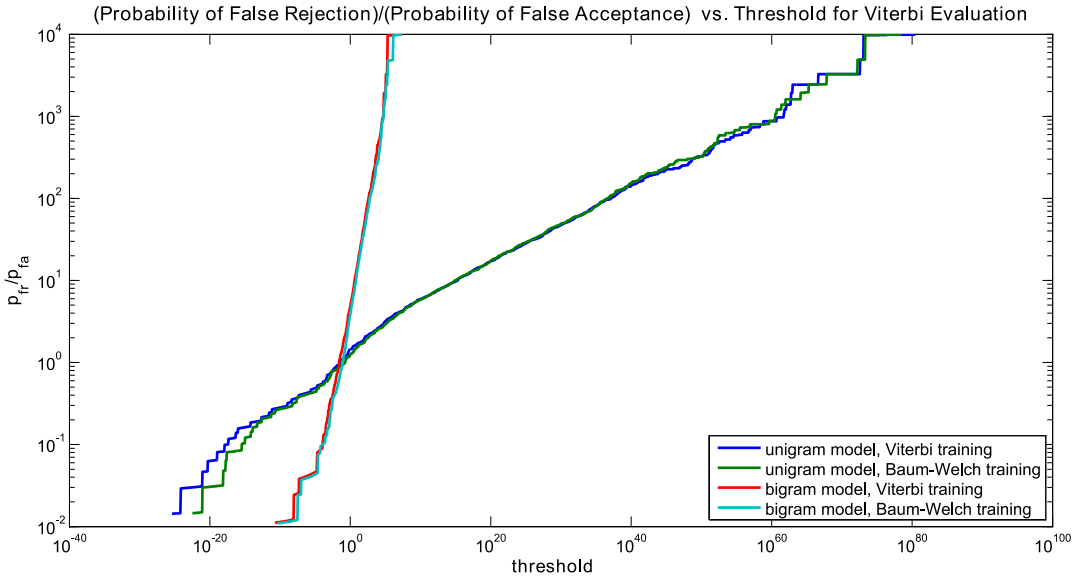

Figure 3.6: Ratio of probability of false rejection and probability of false acceptance for Viterbi evaluation for settings of Figure 3.3

Experiment 2

For this experiment, the following settings have been chosen:

Figure 3.7: Settings for experiment 2

P	D	I	c_A	κ	UBM IDs women	UBM IDs men	reg. women	reg. men	test file ID
8	12	1	1	1	1:20	1:20	51:100	51:100	1

Figure 3.8 shows the performance for models trained in only one iteration. Compared to the results from experiment 1, we can see that the performance for bigram models is much better with multiple training iterations. For unigram models, it does not really seem to matter. Since other experiments have shown the same, five training iterations have been used for the rest of the experiments described in the report.

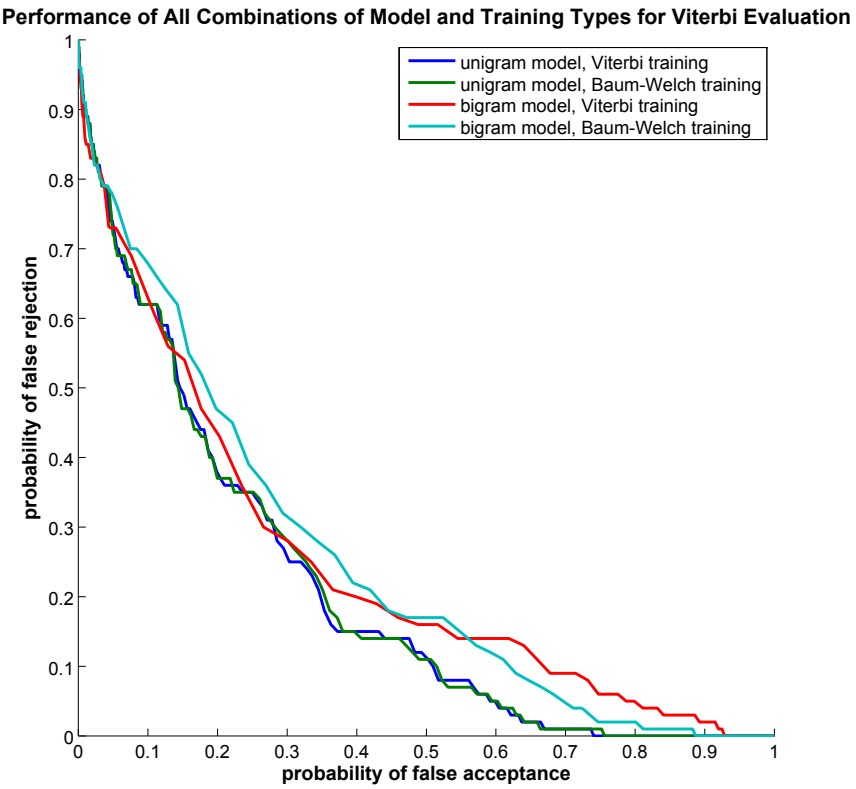

Figure 3.8: Relation between probability of false rejection and probability of false acceptance with Viterbi evaluation for settings of Figure 3.7

Experiment 3

For this experiment, the following settings have been chosen:

Figure 3.9: Settings for experiment 3

P	D	I	c_A	κ	UBM IDs women	UBM IDs men	reg. women	reg. men	test file ID
16	12	5	1	1	1:20	1:20	51:100	51:100	1

As we can see from Figure 3.10, the increase of the number of AAEs (doubled with respect to experiment 1) significantly improved the performance.

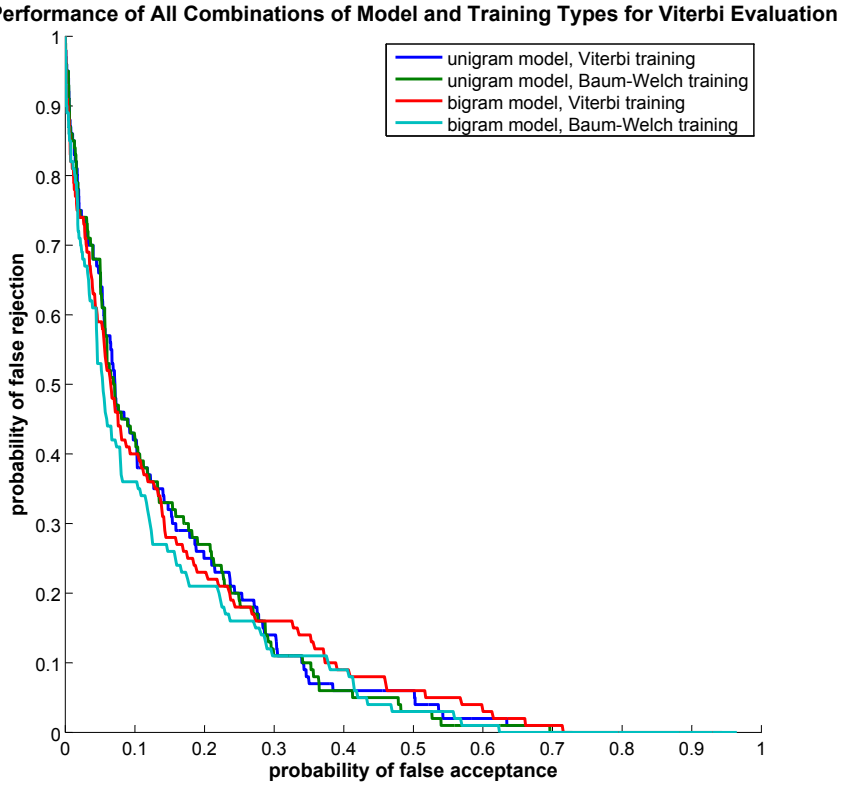

Figure 3.10: Relation between probability of false rejection and probability of false acceptance with Viterbi evaluation for settings of Figure 3.9

Experiment 4

For this experiment, the following settings have been chosen:

Figure 3.11: Settings for experiment 4

P	D	I	c_A	κ	UBM IDs women	UBM IDs men	reg. women	reg. men	test file ID
32	12	5	1	1	1:20	1:20	51:100	51:100	1

For 32 AAEs, the results are again much better (compared to experiment 3) even though the amount of training data is very critical with respect to the number of trained parameters ($\approx$ 1000, cf. Section 2.4.1). A further increase would be interesting, but the evaluation would be computationally very time-consuming.

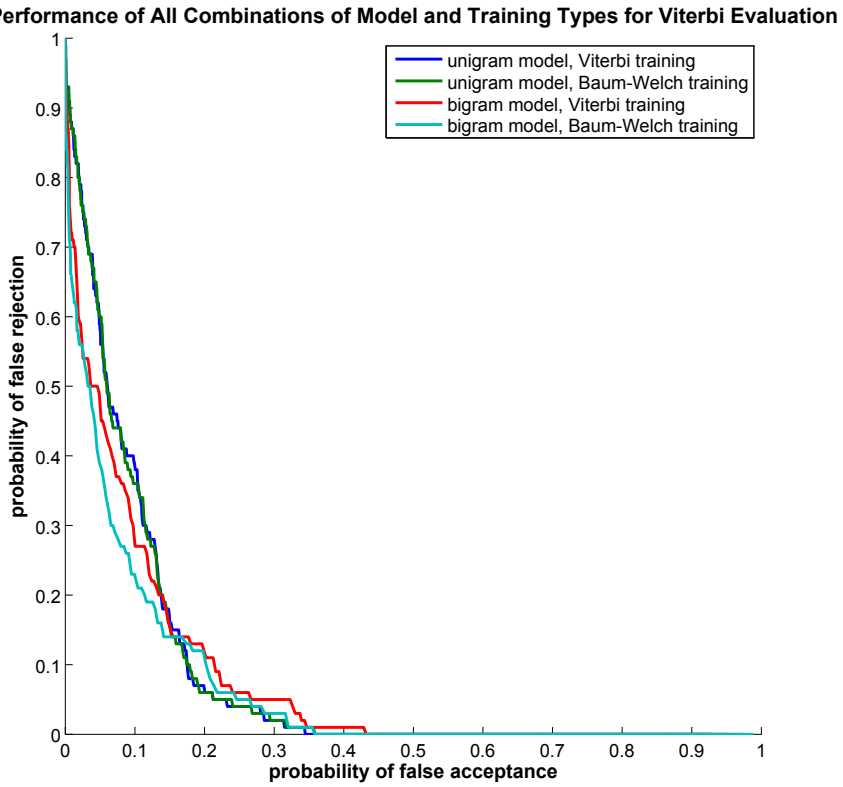

Figure 3.12: Relation between probability of false rejection and probability of false acceptance with Viterbi evaluation for settings of Figure 3.11

Experiment 5

For this experiment, the following settings have been chosen:

Figure 3.13: Settings for experiment 5

P	D	I	c_A	κ	UBM IDs women	UBM IDs men	reg. women	reg. men	test file ID
8	12	5	12	1	1:20	1:20	51:100	51:100	1

The idea of this experiment was to increase the influence of the transition probabilities to the level of influence of the observation probabilities in order to strengthen the speaker-specific characteristics.

Unfortunately, the results are even worse (compared to experiment 1 with $c_A = 1$) as we can see in Figure 3.14, where bigram models seem to be more affected than unigram ones.

Other experiments led to the same results.

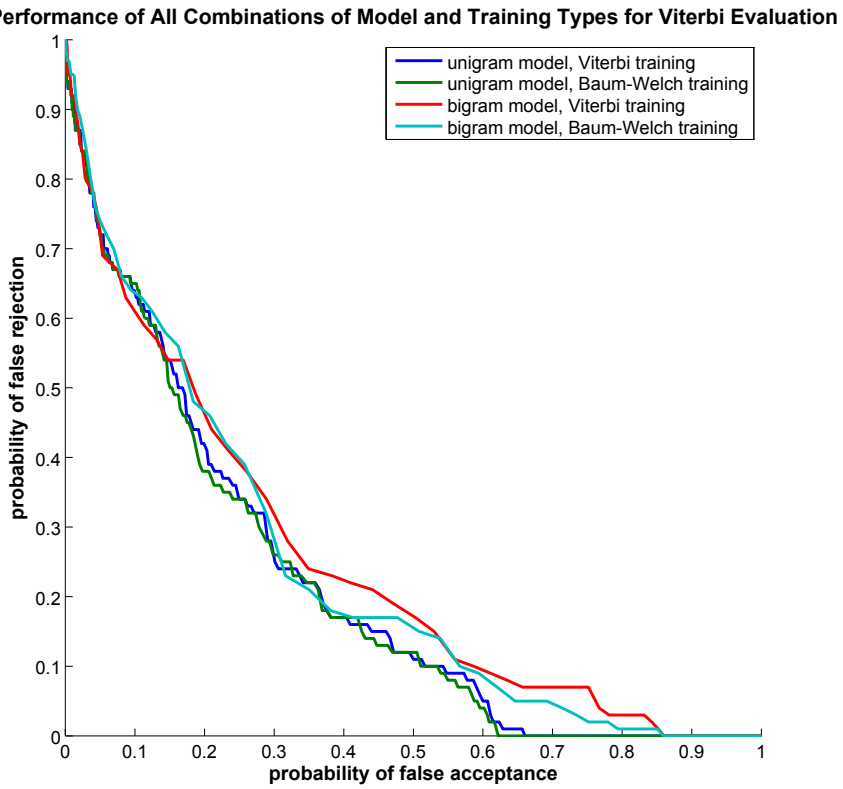

Figure 3.14: Relation between probability of false rejection and probability of false acceptance with Viterbi evaluation for settings of Figure 3.13

Experiment 6

For this experiment, the following settings have been chosen:

Figure 3.15: Settings for experiment 6

P	D	I	c_A	κ	UBM IDs women	UBM IDs men	reg. women	reg. men	test file ID
8	12	5	0.5	1	1:20	1:20	51:100	51:100	1

Since experiment 5 delivered even worse results, the influence of the transition probabilities has been decreased in this experiment. Interestingly, this choice of c_A also worsened the performance for bigram transition probabilities (especially in combination with Viterbi training), but did not really affect the performance for unigram transition probabilities as can be seen in Figure 3.16.

For smaller values of c_A, however, the curves all approach a straight line.

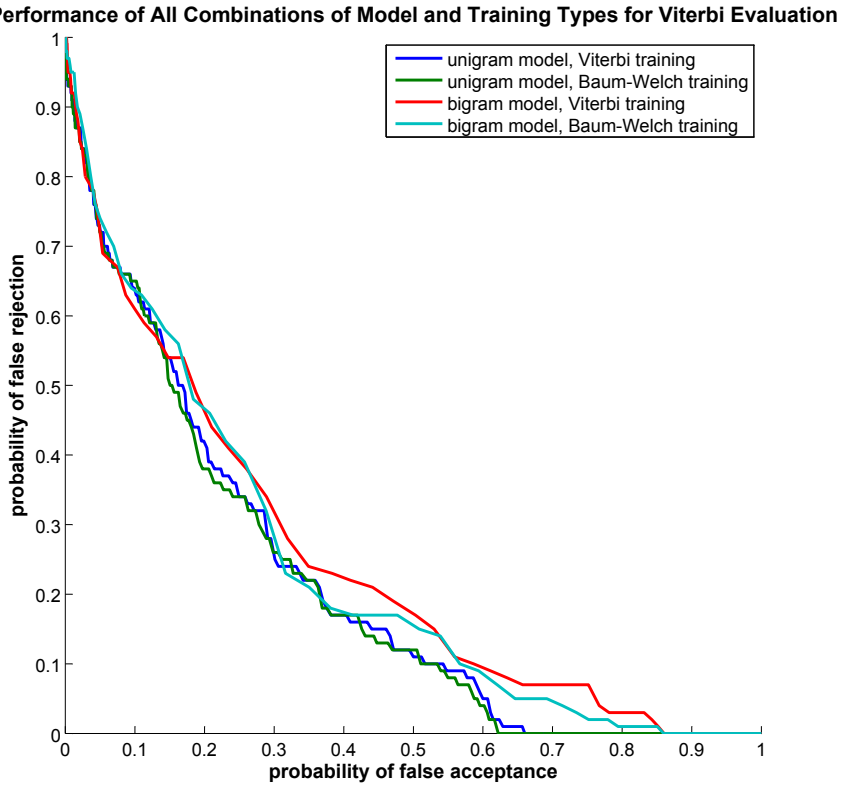

Figure 3.16: Relation between probability of false rejection and probability of false acceptance with Viterbi evaluation for settings of Figure 3.15

Experiment 7

For this experiment, the following settings have been chosen:

Figure 3.17: Settings for experiment 7

P	D	I	c_A	κ	UBM IDs women	UBM IDs men	reg. women	reg. men	test file ID
32	12	5	1	3	1:20	1:20	51:100	51:100	1

In this experiment, the variances of the AAEs have been multiplied by $\kappa = 3$, which should compensate the fact that the variances tend to become very small for higher numbers of AAEs.
This choice of κ seems to affect the performance for bigram models only marginally, but it leads to worse results for unigram models (compared to experiment 4) as we can see in Figure 3.18. The same effect could also be seen for only eight AAEs.

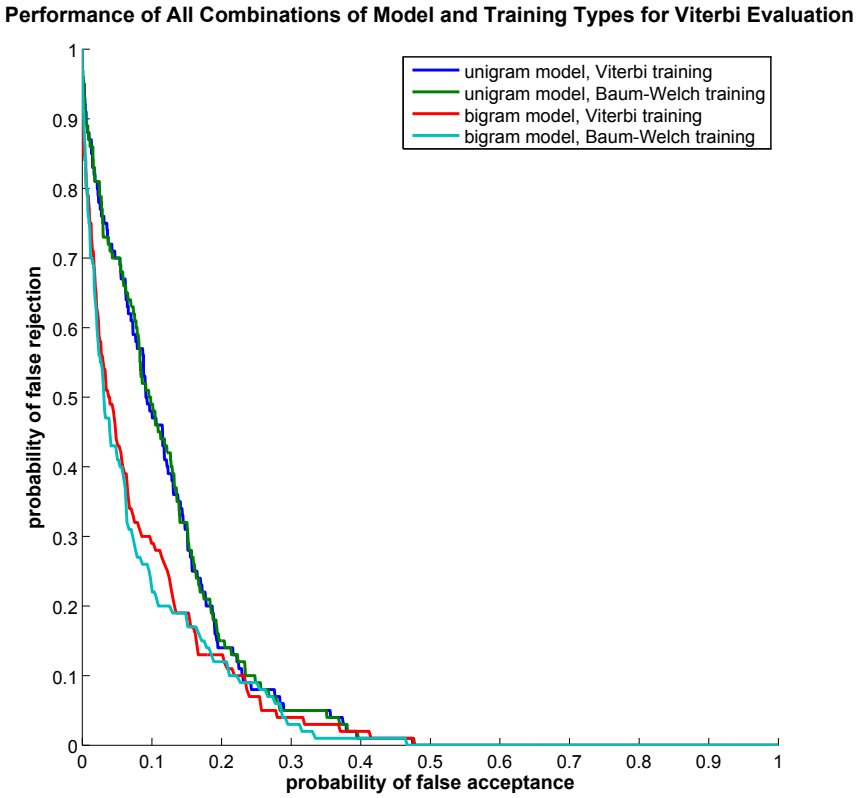

Figure 3.18: Relation between probability of false rejection and probability of false acceptance with Viterbi evaluation for settings of Figure 3.17

Experiment 8

For this experiment, the following settings have been chosen:

Figure 3.19: Settings for experiment 8

P	D	I	c_A	κ	UBM IDs women	UBM IDs men	reg. women	reg. men	test file ID
32	12	5	1	0.5	1:20	1:20	51:100	51:100	1

Finally, the variances of the AAEs have been multiplied by $\kappa = 0.5$ to investigate the effect of doing the opposite of experiment 7.
This time, it seems to be the other way round. The choice of $\kappa = 0.5$ seems to affect the performance for unigram models only marginally, but it leads to worse results for bigram models (compared to experiment 4) as can be seen in Figure 3.20.

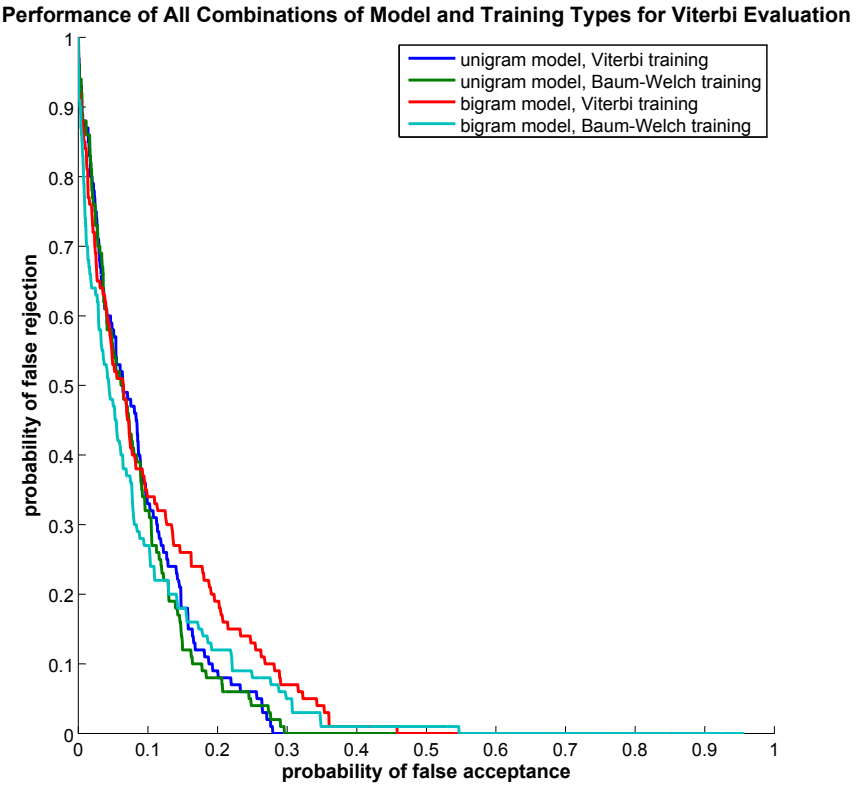

Figure 3.20: Relation between probability of false rejection and probability of false acceptance with Viterbi evaluation for settings of Figure 3.19

3.4.2 Conclusions

- Increasing the number of AAEs is promising with respect to improving the performance, i.e. decreasing the probabilities of false rejection and false acceptance. Probably, this would require more training data for more than 32 AAEs.

  The latter problem would also be important for security applications, were very small values for the probabilities of false rejection and false acceptance are required, i.e. the number of AAEs has to be very large.
- Increasing the influence of the transition probabilities seems not to be a good approach.
- The influence of κ would probably be more interesting for higher numbers of AAEs (e.g. 128).
- For multiple training iterations, $c_A = 1$ and $\kappa = 1$ (optimal setting so far), the bigram models performed better than the unigram ones in the investigated cases. One can even say that the most complex model (bigram transition probabilities in combination with Baum-Welch training) led to the best results even though the performance is very similar for all combinations of model and training types.

Bibliography

[1] B. Pfister and T. Kaufmann. *Sprachverarbeitung: Grundlagen und Methoden der Sprachsynthese und Spracherkennung*. Springer Verlag (ISBN: 978-3-540-75909-6), 2008.

Appendix A

Abbreviations

AAE	abstract acoustic element
GMM	Gaussian mixture model
HMM	hidden Markov model
LR	likelihood ratio
MFCC	mel frequency cepstral coefficient
UBM	universal background model

Appendix B

Implemented Matlab Functions & Scripts

This chapter should give a little overview on the implemented Matlab functions & scripts. For detailed information on a specific function, enter `doc name_of_function` or have a look at the code.

B.1 Functions

baum_welch_training.m

`baum_welch_training.m` estimates the number of times a model is in state S_i for all states S_1 to S_N (ns) and the number of transitions from state S_i to state S_j for all states S_1 to S_N (nt) in order to improve the current transition probabilities of the model (corresponding to a specific speaker or to a UBM) according to the observation sequence X.

compute_LRs.m

`compute_LRs.m` assigns a given speech sample to the most likely one of all registered speakers by computing the logarithms of the likelihood ratios. The production likelihoods are computed with two different (but similar) algorithms:

- Viterbi algorithm
- Forward algorithm

estimate_trans_prob.m

`estimate_trans_prob.m` estimates the transition probabilities between all AAEs starting from an initial transition probability matrix initialized by `init_trans_prob.m`.
For the training, `viterbi_training.m` and `baum_welch_training.m` are used.

extract_MFCCs.m

`extract_MFCCs.m` returns mel frequency cepstral coefficients (MFCCs) of .wav file `filename` and additionally stores the latter as a .mat file.

generate_features.m

`generate_features.m` generates synthetic feature vectors and stores them as .mat files.

generate_model.m

`generate_model.m` computes new transition probabilities for a UBM or a specific speaker. The transition probabilities of a new speaker will be added to the set of already registered speakers.

init_trans_prob.m

`init_trans_prob.m` uniformly initializes a matrix A with transition probabilities for $P = N - 2$ AAEs.

LBG_generate_partitions.m

`LBG_generate_partitions.m` uses the LBG algorithm to partition a D-dimensional (feature) space into P partitions according to a specific training set.

Note: This extended version of `codebookgen.m` of the laboratory exercises requires at least Matlab R2012b because the initialization of the pseudo-random number generator has been adapted to the newest standard.

log_cont_backward_alg.m

`log_cont_backward_alg.m` computes the logarithm of the likelihood that the model given by A, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ produces the observation sequence $\mathbf{X}$. Additionally, it delivers the logarithms of the intermediate joint backward likelihoods.

All computations are done in the logarithmic domain.

log_cont_forward_alg.m

`log_cont_forward_alg.m` computes the logarithm of the likelihood that the model given by A, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ produces the observation sequence $\mathbf{X}$. Additionally, it delivers the logarithms of the intermediate joint forward likelihoods.

All computations are done in the logarithmic domain.

log_cont_viterbi_alg.m

`log_cont_viterbi_alg.m` computes the optimal sequence of AAEs and the logarithm of the likelihood of the given observation sequence along the optimal path (sequence of AAEs).

With this implementation, it is also possible to weight the contribution of the transition probabilities to the likelihood of the observation sequence along the optimal path before the latter is returned.

All computations are done in the logarithmic domain.

logSum.m

`logSum.m` computes the sum $s = x + y$, where x, y and/or s may be very small, so that they can be represented in the log domain only as $\mathrm{logX} = \log(\mathrm{x})$, $\mathrm{logY} = \log(\mathrm{y})$ and $\mathrm{logS} = \log(\mathrm{s})$, respectively. Therefore, it is not possible to compute $\mathrm{logS} = \log(\exp(\mathrm{logX}) + \exp(\mathrm{logY}))$, but it is necessary to apply the Kingsburg-Rayner formula.

Note: Only the description and the logical operator at the beginning have been updated with respect to the version used in the laboratory exercises.

melbankm.m

`melbankm.m` determines matrix for a mel-spaced filterbank.

(unchanged version form laboratory exercises)

mfcc.m

`mfcc.m` evaluates the MFCCs according to the ETSI standard.
(unchanged version form laboratory exercises)

vecquant.m

`vecquant.m` assigns each (feature) vector contained in `vecs` (as row vectors) to the nearest codebook vector contained in `cbk` (as row vectors) by means of the Euclidean distance.
Note: Only the comments and some of the names of the variables have been updated.

verify_speaker.m

`verify_speaker.m` tries to verify a speaker by computing the likelihood ratio (or rather the difference of the logarithms of the likelihoods) of the corresponding speaker-specific model and the UBM for a given speech sample. A speaker is verified if the logarithm of the likelihood ratio exceeds the threshold η.

visualization

`visualization` can be used for two different tasks:

1. Visualization of two specified components of feature vectors (without partitions) (3 input arguments)
2. Visualization of two specified components of feature vectors of a specific speaker or the UBM assigned to the current partitions (7 or 8 input arguments)

viterbi_training.m

`viterbi_training.m` counts the times a model is in each state (ns) and the number of transitions (nt) in order to improve the current transition probabilities of the model (corresponding to a specific speaker or a UBM) according to the observation sequence $\mathbf{X}$.

B.2 Scripts

z_feature_generation.m

`z_feature_generation.m` generates and stores synthetic features for a given number of speakers.

z_model_generation.m

`z_model_generation.m` registers a set of new speakers for the selected training, model and data types.

z_speaker_estimation.m

`z_speaker_estimation.m` estimates the most likely speaker of all registered speakers by computing the likelihood ratios for all speakers and comparing them to each other.

z_speaker_verification.m

`z_speaker_verification.m` tries to verify a speaker by computing its likelihood ratio and comparing it to a given threshold.

zz_TIMIT_auto_eval.m

`zz_TIMIT_auto_eval.m` automatically evaluates speaker verification for the TIMIT data base. All possible parameters can be set at the beginning.

Institut für
Technische Informatik und
Kommunikationsnetze

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo

(SA-2012-32)

Task description for the semester work

of

Mr. Thomas Willi

Main Reader: Dr. Beat Pfister, ETZ D97.6
Tofigh Naghibi, ETZ D97.5

Issue Date: October 22, 2012
Submission Date: December 21, 2012

Text-Independent Speaker Verification with HMM-Based Abstract Acoustic Elements

Introduction

The speech of an individual may be described by means of a hidden Markov model (HMM) as shown in Figure 1. For this purpose, an HMM needs some 50 to 100 states. With each state a mixture of multivariate Gaussian distributions is associated. The high number of parameters of such an HMM requires enough speech material for the training, i.e. at least several hours.

Various methods have been proposed to allow for a sufficiently good training with limited speech material. Most of these methods start from a speaker-independent model that results from the training with enough speech data from many speakers. A speaker-specific model can then be attained through adaptation to a particular speaker which requires only a few minutes speech of that speaker.

In this work a new approach with so-called abstract acoustic elements (AAE) has to be investigated. AAEs have been successfully applied to speech recognition for languages with limited linguistic resources, where speech databases that are large enough for HMM training are missing (see [1]). It has been shown that AAEs can be trained in a speaker- and language-independent manner with speech data from many speakers and from diverse languages. To get the models for the words of the recognition vocabulary, only a few utterances per word are necessary.

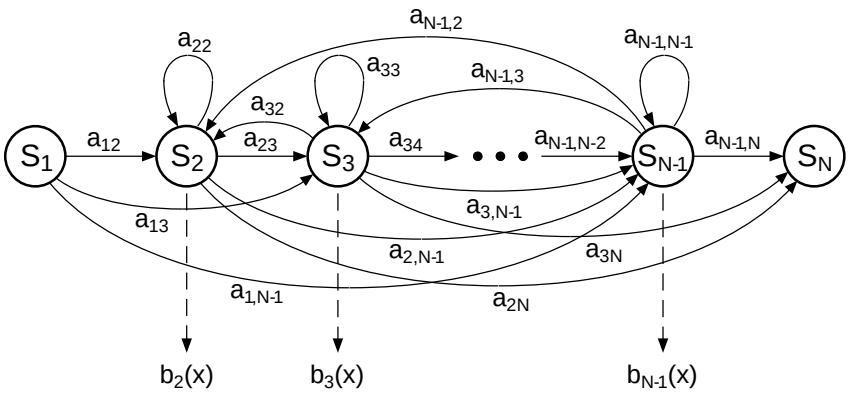

Figure 1: HMM with N states

AAE-based approach to speaker verification

The approach to AAE-based speaker verification is as follows: First, a set of AAE models has to be created. For this we need speech data from many speakers, the so-called training set. Designing a vector quantization from this training set divides the feature space into partitions. The feature vectors of each partition are then used to train a continuous density HMM (one state with one mixture).

Then we need models for individual speakers and a so-called universal background model (UBM). Both types of models have the same architecture: they are basically identical AAE loops with individually trained unigram and/or bigram probabilities (see [2], chapter 13.2.4). Hence, only these unigram and/or bigram probabilities are speaker specific. In other words, we need a set of unigram and/or bigram probabilities for each speaker to be verified and a set of these probabilities for the UBM.

In order to verify a claimed identity from a test utterance, the ratio of the Viterbi probabilities of the corresponding speaker model and the UBM is computed and if it is higher than some threshold, the claimed identity is accepted.

Proposed procedure

In this project, a variety of methods and models has to be applied and optimized for the specific speaker verification task. It is recommended to proceed as follows:

1. Read the chapters about HMM fundamentals, vector quantization and feature extraction in the textbook [2] and perform the associated laboratory exercises.

2. Set up the detailed concept of the AAE-based speaker verification, compile a list of all parameters to be optimized and propose a schedule of the work to realize the speaker verification. Discuss your proposal with the supervisors.

3. Prepare the speech material for your tests. You will need e.g. disjoint data sets for the training of UBM and speaker models, for optimizing system parameters and for test purposes (evaluation of the speaker verification system).

4. Implement the speaker verification preferably in Matlab and reuse code of the laboratory exercises where possible.

5. Evaluate the speaker verification system.

The work done and the attained results have to be documented in a report (see recommendations [3]) that has to be handed in as PDF document. Furthermore, two presentations have to be given: the first one will take place some two weeks after the start of the work and is meant to give a short overview of the task and the initial planning. The second one at the end of the project is expected to present the task, the work done and the achieved results in a sufficiently detailed way. The dates of the presentations will be announced later.

References

[1] M. Gerber. *Speech Recognition Techniques for Languages with Limited Linguistic Resources*. PhD thesis, No. 19507, Computer Engineering and Networks Laboratory, ETH Zurich, 2011.

[2] B. Pfister und T. Kaufmann. *Sprachverarbeitung: Grundlagen und Methoden der Sprachsynthese und Spracherkennung*. Springer Verlag (ISBN: 978-3-540-75909-6), 2008.

[3] B. Pfister. *Richtlinien für das Verfassen des Berichtes zu einer Semester- oder Diplomarbeit*. Institut TIK, ETH Zürich, Februar 2009.
(http://www.tik.ee.ethz.ch/spr/SADA/richtlinien_bericht.pdf).

[4] B. Pfister. *Hinweise für die Präsentation der Semester- oder Diplomarbeit*. Institut TIK, ETH Zürich, März 2004.
(http://www.tik.ee.ethz.ch/spr/SADA/hinweise_praesentation.pdf).

October 29, 2012