TR－SLT－００７２

Speech Recognition for Multiple Non-Native Accent Groups with Speaker-Group Dependent Acoustic Models

Tobias Cincarek, Rainer Gruhn

2004/05/07

概要

In this research, the recognition performance for non-native English speech with different kinds of acoustic models is investigated. The AMs examined are speaker independent native models, speaker-dependent models, AMs built from speech data of non-native accent groups and AMs built from data of speaker clusters. Speaker clusters are derived by k-means clustering in speaker Eigenspace. Oracle experiments and experiments with parallel decoding and hypothesis selection are carried out for performance evaluation.

（株）国際電気通信基礎技術研究所
音声言語コミュニケーション研究所
〒619－0288「けいはんな学研都市」光台二丁目 2 番地 2 TEL：0774－95－1301

Advanced Telecommunication Research Institute International
Spoken Language Translation Research Laboratories
2-2-2 Hikaridai "Keihanna Science City" 619-0288,Japan
Telephone:+81-774-95-1301
Fax :+81-774-95-1308

Contents

Chapter 1

Preface

This technical report is based on the paper draft for the 8th International Conference on Spoken Language Processing (ICSLP) 2004. A software manual and more detailed experimental results are included in this technical report. Additionally, a short survey on acoustic model adaptation will be given in the last chapter. In the last section of that chapter some references to approaches for non-native speech recognition are included.

Work for this technical report, including the collection of speech data from more than 70 non-native speakers, was carried out by Tobias Cincarek and supervised by Rainer Gruhn and Satoshi Nakamura. The author would like to thank Konstantin Markov, for all valuable advice and the provision of a crossword triphone AM for experiments. Further acknowledgments go to Seiichi Yamamoto and Satoshi Nakamura, who made the stay of the author at ATR possible, and all researchers of ATR who helped me with their valuable advice during progress meetings, especially Frank K. Soong.

May 7, 2004, Tobias Cincarek

Chapter 2

An Approach for Non-native ASR

2.1 Abstract

In this research, the recognition performance for non-native English speech with two different kinds of speaker-group-dependent acoustic models is investigated. The approaches for creating speaker groups include knowledge-based grouping of non-native speakers by their first language, and the automatic clustering of speakers. Clustering is based on speaker-dependent acoustic models in speaker Eigenspace. The acoustic model for each speaker group is obtained by bootstrapping with pre-segmented speech data or adaptation of a speaker-independent native baseline model. For the decoding of a non-native speaker's utterance not seen during the training or adaptation phase, the selection of a model suitable to cope with the accent characteristics of that speaker is necessary. Here, ideal selection via an oracle and parallel decoding are examined. Evaluation is conducted in a hotel reservation task for five major accent groups, including German, French, Indonesian, Chinese and Japanese speakers. Recognition results with speaker-dependent and an accent-independent non-native model will also be reported.

2.2 Introduction

Approaches for the recognition of non-native speech found in literature can be classified into three major classes: pronunciation modeling by altering word baseforms, acoustic modeling, and combinations of both. Pronunciation modeling can boost speech recognition performance for a certain foreign accent of the target language, e.g., by adding pronunciation variants to the dictionary, or better applying confusion rules to the phoneme or word lattice during decoding [5] [44]. Such an approach requires either in-depth knowledge of the target language and the first language of the non-native speaker to be able to design pronunciation variants manually, or large amounts of labeled speech data to extract them automatically. Furthermore, that approach covers only deletions, insertions and substitutions of target language phones. To account also for substitutions of target language phones with phones of the non-native speaker's first language, each word in the dictionary can be represented as a sequence of target language phones and a sequence of the non-native speaker's first language phones to form quasi-bilingual models [35] at the same time. An approach for several accents with multi-lingual acoustic models for recognizing digit strings by combining the phone sets and speech data of several native languages was shown to work better than MLLR adaptation of monolingual models with accented speech data [17]. The advantage of bi- and multi-lingual models is the availability of training data from native speech corpora without the need for collecting large amounts of accented data.

However, recent investigations of Flege et al. [19] suggest that non-native speakers may produce speech sounds which are either part of their first language or which were established by merging characteristics of a first language with a target language speech sound. These observations lead to the conclusion that adaptation of each acoustic-phonetic unit model is necessary and that pronunciation modeling with native phone models alone may not be the silver bullet to improve recognition performance for non-native speakers of any accent group.

Approaches which incorporate this phenomenon are the adaptation of a speaker-independent baseline model of the target language [49] or training from scratch with accented speech data [45], the merging of native models [50], and interpolation between native and non-native models [49] or only native models [50]. All these methods lead to a remarkable improvement in recognition performance of foreign accented speech.

Other than the method based on multi-lingual models, the methods summarized here were applied separately for each non-native accent group. To decode an unseen test utterance, the first language of the test speaker needs to be known in order to select an appropriate acoustic model. This step can in principle be carried out by an accent classification system, e.g., realized by an approach based on ergodic HMMs [42], which achieved an accuracy of 65% for six accent groups.

Here we will present a practical system for the recognition of continuous non-native speech of multiple accent groups. For acoustic model selection, parallel decoding with speaker-group-dependent models is employed. Investigation is conducted to determine whether models built with speech data from speakers of knowledge-based speaker groups or models trained with data from data-drivenly created speaker groups are more suitable for recognition. To take account of Flege et al.'s findings, acoustic models are constructed by bootstrapping with pre-segmented non-native speech data. With this approach non-native phones and pronunciation can be learned automatically and coded statistically as HMM mixture distributions.

2.3 Overall approach

The proposed approach consists of an off-line step for the construction of speaker-group-dependent acoustic models and an on-line step for model selection and test utterance decoding. Figures 2.1 and 2.2 illustrate the processing of both steps.

Speaker group formation is necessary for the off-line step. Two methods are investigated: a straightforward knowledge-based approach by grouping the non-native speakers by their first language to build accent-dependent models, and an automatic approach by clustering the speakers in speaker Eigenspace to build cluster-dependent models.

2.4 Speaker clustering

A clustering method in speaker Eigenspace-based on Eigenvoices was introduced in [16] and applied to cluster native speakers. Here the clustering scheme is applied to non-native speakers.

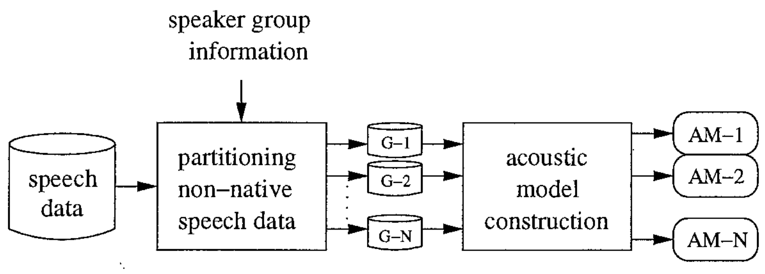
speaker group information
speech data
partitioning non–native speech data
G–1
G–2
G–N
acoustic model construction
AM–1
AM–2
AM–N

Figure 2.1: *Off-line step: Construction of speaker-group-dependent acoustic models. The non-native speech data is partitioned into several training data sets by taking together all the training data from the speakers of each group. From each data set one acoustic model is constructed.*

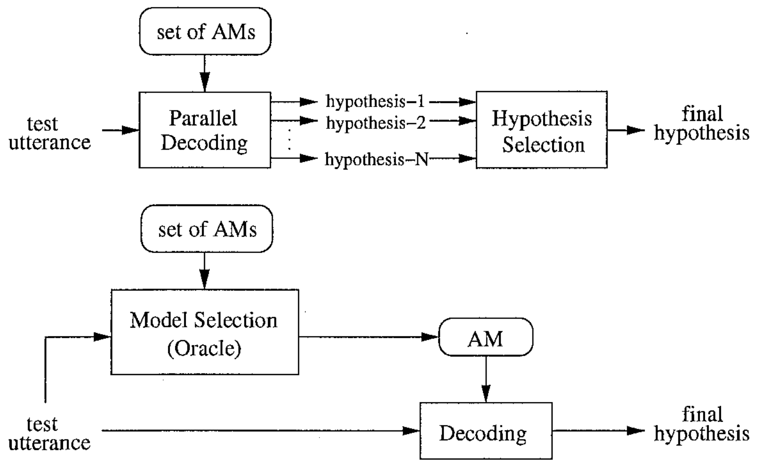

Figure 2.2: *On-line step: Decoding of a test utterance. Top: A test utterance is decoded in parallel with each acoustic model available. The hypothesis with maximum likelihood is selected as the final hypothesis. Bottom: Assuming that a perfect speaker group classification or speaker identification system is available, oracle-based acoustic model selection is simulated to get a reference value for maximum possible performance.*

Eigenspace-based methods have the advantage, that complex representations can be transformed into simpler ones with few parameters while retaining most of the original information.

Speaker-dependent (SD) models were constructed by MAP adaptation of mean vectors of a single mixture speaker-independent monophone model. This adaptation procedure has the advantage, compared with training from scratch, that all remaining model parameters remain the same. Only the mean vectors need to be considered in further processing.

The mean vectors of each SD model were extracted and concatenated to a high-dimensional (39 features $*$ 44 models $*$ 3 states) supervector. Constructing these vectors from GMMs or multiple mixture HMMs would pose some difficulty of alignment between the mixture components of each speaker's model. The correspondence is already given implicitly for single mixture HMMs by model names and state topology.

Principal Component Analysis (PCA) based on the correlation matrix was applied to the supervectors to obtain a basis for an Eigenspace covering most of the sample variance. Finally, the supervectors were projected into this Eigenspace in order to obtain a low-dimensional representative for each speaker, which is suitable for clustering.

As clustering algorithms, k-means and agglomerative hierarchical clustering with four different kinds of inter-cluster distance measures as described in [12] were examined. Inter-vector distances were measured by the Euclidean distance. For the k-means algorithm, each cluster was initialized with the speakers of each first language group.

2.5 Non-native speech database

Read speech data of about 100 non-native English speakers was collected at ATR. It is clean speech recorded at 16-kHz sampling frequency and 16-bit precision. The data consists of 48 phonetically balanced sentences of the TIMIT set and six hotel reservation dialogs. To be able to abstract from variability introduced by gender, only the data of 75 non-native male speakers was actually used. The speaker set utilized for experiments consists of 15 Japanese, 15 Chinese, 15 French, 15 German and 15 Indonesian natives. All speakers utter the same sentences. The training and adaptation data set comprises 88 utterances ($\approx$ 10 minutes), the validation data set ten utterances ($\approx$ 1 minute) and the test data set 23 utterances ($\approx$ 3 minutes) per speaker.

For comparison of recognition performance of natives vs. non-natives, speech data of six native English speakers uttering the same test sentences as the non-native speakers was used.

2.6 Baseline system

HTK was employed for training and adaptation of all acoustic models, building of the language model and decoding in all evaluation experiments. The configuration of the baseline system is as follows:

2.6.1 Acoustic model

More than 60 hours (37,413 utterances) of speech data from the LDC Wall Street Journal corpus (WSJ) were used to build three speaker-independent native English acoustics with different complexity:

1. 44 Monophone 3-state HMMs with 16 mixtures
2. State-clustered biphone model with about 3,000 states and 10 mixtures
3. State-clustered crossword triphone model with about 9,600 states and 12 mixtures

39 acoustic features, 12 MFCC coefficients and energy with first and second derivation, were extracted every 10 ms. The word accuracy of these three acoustic models on the Hub2 5K evaluation task was 80.8% for the monophone, 86.8% for the biphone and 93.6% for the triphone model.

Since only speech data from male non-native speakers are considered in this research a gender-dependent monophone model was built by MAP adaptation of the SI baseline AM with the speech data of all male speakers from WSJ.

2.6.2 Language model

The n-gram probabilities were estimated from a database with 235 dialogs in the hotel reservation domain comprising 6,460 utterances with 65,893 words in total. The lexicon contained about 8,800 entries for about 7,300 words including compounds. The perplexity for the 344-word evaluation task, two dialogs with 23 utterances in total, was 32.

2.7 Results

For evaluation, 75-fold leave-one-speaker-out cross validation was carried out for all experiments with speaker-group-dependent models in order to obtain a realistic estimate of performance. Speaker-group-dependent models consist of 42 HMMs with ten mixtures. In each table the average word accuracy for each speaker group is shown. Initially three approaches for construction of the accent-dependent models were examined. Performance was best for bootstrapping the models with pre-segmented non-native speech data obtained by forced-alignment with the monophone SI native baseline model. Results were slightly lower for training the models from scratch, followed by MAP adaptation of the native monophone SI baseline.

2.7.1 Speaker-independent models

Recognition accuracy with the speaker-independent baseline system as described in section 2.6 varies remarkably for each acoustic model and speaker group. For native speakers there was an increase, or at least no decrease in accuracy on average when decoding with the biphone and the triphone model in comparison to the monophone model. However, for non-native speech severe degradations can be observed (see Table 2.1). While the relative drop in accuracy was rather low for German speakers with about 8%, error rates almost doubled for Japanese speakers. This may be due to phonetic errors and different coarticulation of speech sounds especially for speakers whose first language is Japanese and Chinese, who have fewer speech sounds in common with the English language than German, French or Indonesian. A comparison of IPA-based [1] phone sets revealed that German has at least 28, Indonesian 26, French 25, Mandarin Chinese 21 and Japanese 19 phones in common with American English. In further experiments monophone models are employed, because they are more robust to accent variability and require less data for training and adaptation than context-dependent models.

Table 2.1: *Recognition performance with the speaker-independent (SI) native English baseline acoustic models.*

Model	Eng	Ger	Fre	Ind	Jap	Chi
SI mono	82.4	75.7	71.9	70.7	55.4	63.3
SI biph	82.5	72.5	66.9	63.1	42.9	54.9
SI triph	85.5	69.6	62.1	51.7	31.4	39.0

2.7.2 Speaker clustering

Several clustering methods with different distance measures were examined. Hierarchical clustering produced rather balanced clusters for the furthest neighbor distance but rather sparse clusters for the centroid distance and average inter-vector distance, and very sparse clusters for the nearest neighbor metric. The tendency of producing sparse clusters also increased with the dimension of the Eigenspace. Since clusters generated by the k-means algorithm were more balanced, even when setting the Eigenspace dimension to 20, being equivalent to capturing nearly 95% of sample variance, their speaker configurations were used for building the cluster-dependent models. The distribution of speakers' first languages in each cluster is shown in table 2.2.

Table 2.2: *Distribution of non-native speakers in clusters created by k-means clustering in a 20-dimensional Eigenspace.*

Cluster	1	2	3	4	5
Chinese	-	4	2	5	4
French	7	5	-	3	-
German	3	8	-	1	3
Indonesian	-	2	-	3	10
Japanese	-	-	13	1	1

2.7.3 Oracle experiments

Knowing the first language, cluster membership or identity of the test speaker, several oracle-based experiments for obtaining reference values for maximum recognition performance can be carried out. Results are summarized in Table 2.3.

An estimate of maximum possible performance for each non-native speaker can be obtained by decoding with speaker-dependent (SD) models. There is a remarkable increase in accuracy for all non-native speakers in comparison to the gender-dependent baseline model. The performance with SD models for six native English speakers was 92.6%, indicating that non-native speech is indeed more variable than native speech.

The performance with accent-dependent (AD) models is also high, suggesting that the accent characteristics of speakers having the first language in common are similar. The difference in accuracy between SD and AD models is largest for the Chinese speaker group, which can be explained by the fact that this group consists of speakers from several areas of China, also including some speakers whose first language is Cantonese.

Recognition with cluster-dependent (CLD) models still leads to good performance, but is slightly lower than that with accent-dependent models.

2.7.4 Parallel decoding

In order to build a practical ASR system for non-native speech recognition, parallel decoding with the accent-dependent or the cluster-dependent models was employed. This procedure yields one

Table 2.3: *Recognition performance with the gender-dependent (GD) native baseline monophone model, the speaker-dependent, the accent-dependent and the cluster-dependent models.*

Model	Ger	Fre	Ind	Jap	Chi
GD baseline	75.7	73.8	71.9	55.6	63.7
Cluster-dep.	80.1	82.8	82.9	82.1	75.5
Accent-dep.	82.7	84.4	85.4	82.2	77.3
Speaker-dep.	87.2	87.5	87.7	84.6	82.8

Table 2.4: *Non-native speech recognition performance by parallel decoding with accent-dependent (AD) or cluster-dependent (CLD) models.*

Model	Ger	Fre	Ind	Jap	Chi
AD parallel	80.6	83.6	83.0	80.4	75.5
CLD parallel	80.1	82.8	82.9	82.0	75.5

recognition hypothesis from each acoustic model for an unseen utterance. The hypothesis with maximum acoustic likelihood was selected as the final recognition result.

The results for parallel decoding are summarized in Table 2.4. There is a small drop in recognition accuracy in comparison to the oracle experiment of section 2.7.3 for the accent-dependent models, but no performance decrease for the cluster-dependent models. Furthermore, both model types yield better results than the GD baseline. The difference in accuracy between AD and CLD models is significant for the Japanese speaker group only.

While the cluster classification accuracy (64.6%) was higher than the accent classification accuracy (52.5%), parallel decoding with CLD models may in practice perform better than decoding with AD models if data of more speakers become available. The results for each speaker group are summarized in Figure 2.3.

2.7.5 Non-native model

To investigate whether the pronunciation variations of all speakers from the considered five accent groups can be captured by only one monophone acoustic model, the data of 50 non-native speakers, 10 from each accent group, were used to train one 16 mixture non-native monophone model (NN). Evaluation was carried out with the remaining 25 speakers, doing 3-fold cross-validation. The speakers for each training and test set were randomly selected, taking care that the first languages of speakers in training and test sets were distributed uniformly.

Table 2.5: *Performance with a non-native monophone model.*

Model	Ger	Fre	Ind	Jap	Chi
SI NN	80.5	83.1	83.2	79.7	79.8

As Table 2.5 illustrates, the performance is almost equal to parallel decoding with AD or CLD models, except for the rather accent-inhomogeneous Chinese speaker group, which may be due

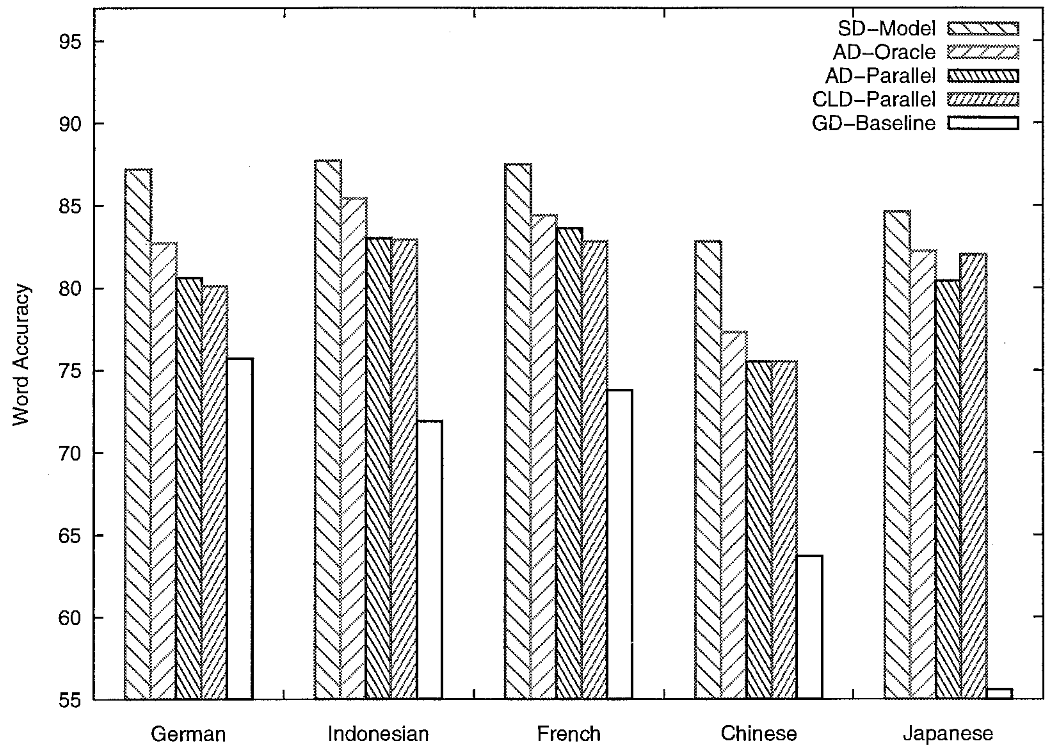

Figure 2.3: *Comparison of experimental results.*

to the higher robustness of the non-native model, which was trained with 50 speakers, than the AD and CLD models, for which data of only 15 speakers were available. Since accuracy with AD models in the oracle experiments of section 2.7.3 is still significantly higher than that with the NN model, accent-dependent models may in principle perform better than accent-independent models, whenever an accent classification system with high accuracy is available.

2.8 Conclusion

A practical approach for non-native ASR was introduced. It is based on parallel decoding with several speaker-group-dependent monophone acoustic models and maximum likelihood hypothesis selection. Good accuracy with accent-dependent models was achieved for five non-native accents groups with a relative improvement of 6% up to 44% on average to a GD native baseline depending on the speaker group.

The maximum recognition performance with monophone models is limited. However, as long as large corpora of non-native speech are not available, training of robust context-dependent acoustic models is infeasible. Assuming rather consistent pronunciation variations of non-native speakers within each accent group, higher accuracy may be possible with accent- and context-dependent models.

Chapter 3

Software

This chapter explains the software which was written for carrying out the experiments in this research. Tools and scripts for acoustic model training and adaptation with HTK, training and test data partitioning, parameter extraction from HTK AM files, Eigenvoice computation, speaker clustering, reading and writing HTK MLF files, etc. will be described. The scripts are programmed in and were tested for RUBY V1.6.8 (.rb extension), PYTHON V2.3.2 (.py extension) or PERL V5.6.1 (.pl extension). There is a short help message for most scripts, when it is executed on the command line without any options. Almost every script uses publicly available libraries and libraries written by the author. These libraries are summarized in Table 3.1. Examples for how to use many of the scripts described here, can be found in chapter 4. Even more examples are available in the `scripts` directories of each experiment.

Table 3.1: *RUBY and PYTHON software libraries.*

ID	Lib-Name	Author/Status	Purpose
L1	uci.rb	T.Cincarek	Command line parsing
L2	misc.rb		Miscellanenous functions
L3	htk2.rb		HTK tools interface
L4	matvec.rb		Matrix and vector functions; i-face to L6
L5	log.rb		Simple logging facility
L6	matrix-algebra.rb	S.Hara	Matrix algebra
L7	numarray.py	Open Source	Matrix algebra V0.7
L8	vq.py	T.Cincarek	Vector quantization / Clustering
L9	stats.py		Mean, Co-Variance, Correlation, ...
L10	pca.py		PCA based on L7 and L9
L11	misc.py		Miscellaneous functions
L12	spkdata.rb		SLT Non-native DB speaker information

3.1 HTK Training and Adaptation

The Hidden Markov Model Toolkit (HTK) provides tools for acoustic model training and adaptation. The procedure for training and adaptation follows the descriptions given in the HTK handbook. Several scripts employing HTK tools were written:

ID	Script	Purpose
1	htk-buildmonophAM.rb	Monophone AM training from scratch
2	htk-bootstrapmonophAM.rb	Monophone AM training with pre-segmented data
3	htk-buildbiphoneAM.rb	Biphone AM training, begin with 1-mix monophone AM
4	htk-makelm.rb	Build language model
5	htk-adapt.rb	MLLR and/or MAP AM adaptation
6	htk-fex.rb	acoustic feature extraction
7	htk-genproto.rb	Generate HMM prototype
8	htk-genproto2.rb	Generate HMM prototype
9	htk-cloneproto.rb	Clone prototype HMM for each phoneme
10	htk-fixsilence.rb	Fix silence/pause HMM

Script (1) uses scripts 7,9,10, script (2) uses scripts 7,8,10. Scripts 1,2,3 use functions especially from library (L3). All scripts should work for HTK versions 2.2, 3.0, 3.1 and 3.2.

3.2 Training, Validation and Test Data Set Partitioning

The script `listsets.rb` can be used for splitting up the non-native speech data into (speaker-disjoint) training and test data sets. Separate file lists (train, vali, test) can be generated for each group of non-native speakers, i.e. first language groups or speaker clusters. There must be a `<source>` directory containing the `<spkid>.{adapt,eval,vali}.mfcc` files. These file extensions can be changed by editing the script itself. For example, a typical command line call looks like this:

```
listsets.rb -c -d -l <langlist> -t <target>  -b <source> -s <spklist>
listsets.rb -c -d -l C,F,G,I,J  -t targetdir -b lists    -s M001
```

This call

- `-c` deletes any `list.*` file in `-t <target>` directory
- `-d` generates speaker-disjoint training and test data sets
- concatenates all the files `<spk>.adapt.mfcc` from `-b <source>` directory to `list.train.<lang>` and all the files `<spk>.vali.mfcc` to `list.vali.<lang>` of any speaker, whose first language is in `-l <langlist>`;
- the same is done with all `<spk>.eval.mfcc` in order to generate file `list.test.<lang>`, excluding the files of speakers whose ID is in `-s <spklist>`.

The information about each speaker's first language (of the ATR SLT non-native English DB) is stored in the RUBY library file `spkdata.rb`. For using different speaker information, an appropriate file can be specified with option `-g <spkinfo>`. Instead of language information, speaker cluster information can be included. A speaker information file has one line per speaker. Each line contains the speaker ID, the speaker's first language (or cluster or group ID), the speaker's age and the speaker's score separated by white space. However, only the information about each speaker's first language, group or cluster is used by this and most other scripts.

3.3 HTK AM Parameter Extraction

The script `htk-extractvoices.rb` can be used to extract acoustic model parameters, i.e. mixture mean vectors, mixture variance vectors, g-constant and mixture weights, from an HTK macro file. The macro file must be in ASCII format. Furthermore, any model with tied parameters must first be converted into an untied model before using this script. Usage example:

```
htk-extractvoices.rb -h <hmmfile> -p <hmmlist>    -s <statelist>
htk-extractvoices.rb -h hmm.mono  -p ax,axr,aw,ao -s 2,3
```

This call extract the parameters of the states two and three of the HMMs "ax", "axr", "aw" and "ao". If no list of states of no list of HMMs is given, the parameters of all states and all models are extracted respectively. Output is in ASCII and has the following format:

```
<HMM-Name> <State-Name> <Num-Mixes> <Num-Feats>
wts <weights>
gc <gconstant 1>
mean <mean-vector 1>
var <variance-vector 1>
gc <gconstant 2>
mean <mean-vector 2>
var <variance-vector 2>
...
<HMM-Name> <State-Name> <Num-Mixes> <Num-Feats>
...
```

This ASCII output can be converted into binary format with `modasc2bin.py`. Conversion from binary to ASCII is possible with `modbin2asc.py`. See script implementation for details of the binary format. The binary I/O is based on the PYTHON library `numarray`. These two conversion scripts read input from STDIN and write output STDOUT.

3.4 Eigenvoice Computation

Eigenvoices can be calculated with the script `eigenvoices.py`. Before using this script, HMM parameters have to be extracted with `htk-extractvoices.rb` and converted to binary format with `modasc2bin.py`. This has to be carried out for every speaker. One parameter file for each speaker with the filename <spkid>_<name>.params (containing the supervectors) must be generated and copied to the same directory <dir>. Finally, Eigenvoices can be computed by executing

```
eigenvoices.py -i <spklist> -p <name> -s <dir>    -t <target>
eigenvoices.py -i spklist   -p all    -s paramdir -t evdir
```

The file <spklist> must contain the speaker IDs (one per line) for all speakers' models' parameters which should be included in the Eigenvoice computation. The script eigenvoices.py only makes use of the mean vectors. Further parameters, i.e. variance vectors and mixture weights could be included by editing this script. The script produces the files eigenvalues_<name>, eigenvectors_<name> and eigenmodels_<name> (Eigenvoices) in the <target> directory. All output files are in ASCII. The files eigenmodels_<name> and eigenvectors_<name> have the following format:

```
<Number_Eigenvoices> <Eigenvoice_Dimension>
<Eigenvoice-Vector 1>
<Eigenvoice-Vector 2>
...
```

The file eigenvalues_<name> contains one Eigenvalue per line in descending order (of magnitude). If the number of input vectors to the script is N, the eigenvectors_<name> file contains N+1 vectors, i.e. the mean vector of all original supervectors $\vec{\mu}$ and the first N Eigenvectors. The file eigenmodels_<name> contains N "model vectors". These model vectors were obtained by subtracting the mean vector $\vec{\mu}$ from the original supervector $\vec{m}^j$ of speaker j and then projecting it into the Eigenspace. Each component c_i^j of the resulting model vectors is calculated via Equation 3.1.

$$c_i^j = \vec{e}_i^T(\vec{m}^j - \vec{\mu}) \tag{3.1}$$

$\vec{e}_i$ is the Eigenvector of the i-th largest Eigenvalue. Eigenvector computation in eigenvoices.rb is based on the correlation matrix of the supervectors $\vec{m}^j$. Computation based on the covariance matrix is possible by using pca.pca_cov instead of pca.pca_cor of library L10.

3.5 Speaker Clustering in Eigenspace

Speaker clustering can be carried out with the script clusterspk.py. This script uses routines from the PYTHON libraries L8, L9 and L11. Most important is L8 with the clustering algorithms. The k-means clustering algorithm and agglomerative hierarchical clustering with four kinds of inter-cluster distances are implemented. Inter-vector distances are computed by the Euclidean distance between sample vectors. Each speaker is expressed as one "model vector". The calculation of this vector was already described in the previous sections. The script clusterspk.py needs the file -m eigenmodels_<name> from eigenvoices.py. That file contains the representative vector for each speaker. The information of all speakers must also be provided with option -s. The order of speakers in that file must be the same as in the model file. Consequently, the same speaker information file should be used for both scripts.

K-means. The `-a k-means` algorithm is an unsupervised clustering algorithm. It is supposed to find a previously determined, fixed number of clusters. The number of clusters can be specified by option `-c`. The number of iterations can be is set by option `-i`. However, the k-means algorithm terminates, if the VQ-distortion between the last and the current iteration decreased less than `thres = 0.00001`. Since the algorithm belongs to the EM (expectation maximization) family, initialization is necessary, i.e. initial clusters have to be set up before the clustering algorithm can be iterated. The default behavior of script `clusterspk.py` is to initialize each cluster with one sample, which is randomly selected from the whole sample set by picking each n/k-th sample, if there are n samples and k clusters. Knowledge-based initialization of clusters is possible with option `-k`. Each cluster is then initialized with the speakers of each speaker group. Speaker groups are defined in the `-s <spkinfo>` speaker information file. There must be more speaker groups than `-c <clusters>` if knowledge-based initialization is employed.

Hierarchical clustering. Agglomerative hierarchical clustering with four different kinds of inter-cluster distances is implemented. It is a bottom-up clustering scheme, i.e. in the beginning there are as many clusters as samples. The two clusters, which are nearest to each other with respect to the inter-vector and inter-cluster distance measure are merged successively until the desired number of clusters is reached. The inter-cluster distance measure has to be selected by `-a <algo>` together with the algorithm ID. The `<algo>` IDs and the corresponding inter-cluster distances with their definitions are as follows:

- nearest neighbor, option `-a h-min` $d(A,B) = \min_{s\in A, t\in B} ||\vec{v}_s - \vec{v}_t||_E$
- furthest neighbor, option `-a h-max` $d(A,B) = \max_{s\in A, t\in B} ||\vec{v}_s - \vec{v}_t||_E$
- average distance, option `-a h-avg` $d(A,B) = \frac{1}{|A||B|} \sum_{s\in A, t\in B} ||\vec{v}_s - \vec{v}_t||_E$
- mean distance, option `-a h-mean` $d(A,B) = ||\frac{1}{|A|} \sum_{s\in A} \vec{v}_s - \frac{1}{|B|} \sum_{t\in B} \vec{v}_t||_E$

The symbols A and B each represent a cluster, i.e. a set of speakers. All symbols $\vec{v}$ represent each speaker's representative vector. $|A|$ means the number of vectors in cluster A. $||\vec{a} - \vec{b}||_E$ is the Euclidean distance between vectors $\vec{a}$ and $\vec{b}$.

It is expected, that speaker IDs begin with "M" or "F", indicating a speaker's gender. Then it is possible to select only all male or only all female speakers with option `-t {M,F}`.

The component range of the "model vector" which should be used for distance computation during clustering can be specified by option `-f <begin>,<end>`.

Results of clustering, i.e. the speaker cluster information, can be written to a separate speaker information file with option `-g`. Furthermore, the merging history for hierarchical clustering can be saved into a separate file with `-h <hfile>`.

3.6 HTK MLF Input/Output

Library L3 contains the class `MLF` for reading and writing HTK master label files (MLF). This I/O interface is necessary for parallel decoding experiments, where the recognition output files for several acoustic models must be parsed to be able to select the appropriate recognition hypothesis. The selected hypothesis can again be written to a master label file. Important methods of the MLF class are summarized in Table 3.2.

Table 3.2: *Methods of the MLF class.*

Method Name	Return value	Arguments	Function
initialize (constructor)	object	1: filename 2: feature string	Name of MLF file to open Contents of MLF file, i.e. columns
read	data hash	1: utterance ID 2: ID suffix	(sub)string of utterance IDs to read suffixes of utterance IDs in MLF
read_TW	data hash	1: utterance ID 2: ID suffix	Arguments are the same as for read(). But only columns T,W are present in MLF. Moreover, T data is discarded.
read_W	data hash	1: utterance ID 2: ID suffix	Arguments are the same as for read(). However, only W column is present in the source MLF.
write	none	1: ID suffix 2: filename	ID suffixes in target file Name of target file to write
get_data	utterance data (list)	1: utterance ID	Get data of utterance, i.e. list of segment times, segment labels, ...
set_data	none	1: utterance ID 2: utterance data	Set data of utterance in MLF object

The "feature string" can be made up of the characters "T" (start and end time), "W" (word label), "P" (phone label) and "L" (likelihood score). The order of characters in this string define the order of columns in the label file to read. The keys of the "data hash" are the utterance IDs without suffix, e.g. ".rec" or ".lab". Each hash entry is either a list with the word or phonemes sequence of the utterance, or a list of lists, which each list containing the information of each utterance segment. The order of items in this list is the same as in the MLF.

3.7 HTK MLF Processing

There are several scripts for converting MLFs or extracting information from MLFs. Table 3.4 gives an overview.

The three scripts {11,12,13} read input from STDIN and write results to STDOUT. Scripts {11,12} are straightforward to use. Further explanation will be given for scripts {13,14} only.

The main script for computing AM and LM scores is `mlf-score.rb`. Input is an MLF with one or more utterances with segment start and end times, word or phoneme labels, and acoustic scores. A list of allowed phoneme or word tokens must also be specified with option `-p`. Output is one line per utterance with one AM, both AM and LM, or one combined AM and LM score. Two types of acoustic scores can be calculated: The total acoustic likelihood score `-s`, which is obtained by just summing up all segment scores, or the average segment score `-m`, which is obtained by dividing the total acoustic score with the number of phoneme or word segments in

Table 3.3: *RUBY scripts for MLF file processing.*

ID	Script name	Purpose
11	mlf-stripus.rb	Conversion of compound words with underscores and apostrophes into separate words, e.g., “PHONE_NUMBER” to “PHONE” and “NUMBER”, or “I'M” to “I” and “M”
12	mlf-strippath.rb	Convert utterance ID with path in MLF to “*/filename”
13	mlf-score.rb	Extraction of the utterance-level AM scores from MLF and Combination with LM score, which can be calculated by script (14).
14	htk-lmscore.rb	Calculaton of LM score with HTK LM tools

the utterance. The segment scores can additionally be normalized by the duration with option `-n`. Segment labels, whose scores should be ignored, can be specified by option `-d`.

For calculating LM scores, a bigram statistics file in ARPA/MIT-LL format must be provided with option `-b`. The weighting of the LM score against the AM score can be determined by option `-w`. If `-w` is not specified, AM and LM score will be output separately. The LM score computation is done by an internal call of script (14), which is based on the HTK LM tool `LPlex`.

However, in order to obtain utterance-level acoustic likelihood and language model scores it is better to use the scores which are written to STDOUT by HVite during the decoding or alignment process. From this output stream AM and LM scores can be extracted by piping it through the script `htk-vitestdoutparse.rb`.

3.8 Miscellaneous Scripts

In Table 3.4 several important and often used scripts are explained briefly. Scripts {16,17,18,19} are designed to operate on text streams and are straightforward to use. Scripts {20,21} are for multi-column score streams from several classifiers. These two scripts determine for each score line read from STDIN the column with the smallest (`mincol.pl`) or the largest (`maxcol.pl`) score respectively and write the column index together with the input scores to STDOUT.

Table 3.4: *Miscellaneous scripts.*

ID	Script name	Purpose
15	rectable.rb	show recognition rates for each speaker group
16	words.rb	tokenize word strings
17	stripnl.rb	concatenate word strings
18	upcase.rb	capitalize letters in input stream
19	downcase.rb	convert capital to small letters
20	maxcol.pl	*argmax*-operator
21	mincol.pl	*argmin*-operator
22	recrate.pl	calculates and displays recognition rates for a score file or a result file

The script `rectable.rb` is used calculate the mean word accuracy or correct rate together with its minimum, maximum and standard deviation for several speaker groups or speaker clusters. Input to the script is a `-r <result>` file with the following format:

```
<SpkID 1>: [<LangID/GroupID>] WORD: %Corr=##.##, Acc=##.## [H=###,...,N=###]
<SpkID 2>: [<LangID/GroupID>] WORD: %Corr=##.##, Acc=##.## [H=###,...,N=###]
...
```

Everything from string "WORD:" on can be obtained by executing the HTK recognition rate calculation tool "HResults" for a file which contains the recognition result for speaker with `<SpkID>`, and then `grep` the string "WORD:". If `<LangID>` or `<GroupID>` is registered is library `spkdata.rb`, it will be expanded, e.g. language name "German" for language ID "G".

The script `recrate.pl` can be used to calculate the recognition rate for *argmax* or *argmin* classification of the score output of N classifiers for N classes, e.g. one GMM score or HMM score per class. A score file contains N scores per line, separated by white space. A result file additionally contains the class index corresponding to the row of the classifier with the highest/lowest score as the first entry. Rows are indexed with integers from 1 to N.

```
<ClassID> <Score1> <Score2> ... <ScoreN>
<ClassID> <Score1> <Score2> ... <ScoreN>
...
```

For example, if the files `scores.A` and `scores.B` contain the scores for the classification of samples of classes A and B respectively with both classifier models A and B, the recognition rate (total and class average) can be calculated and displayed by calling `recrate.pl` with options `-s scores`, `-r result` and `-l A,B`.

3.9 HTK Configuration

```
SOURCEFORMAT = HTK
SOURCEKIND   = MFCC_E_D_A_Z
SOURCERATE   = 100000
WINDOWSIZE   = 200000
NUMCEPS      = 12
```

3.10 Environment Variables

In order to use software libraries, which are installed locally or were written by the author, the following environment variables must be set to the path names in which the libraries reside.

```
setenv RUBYLIB /home/xtcinca/mylib/ruby
setenv PYTHONPATH /home/xtcinca/mylib/python:/home/xtcinca/lib/python
setenv PERLLIB /home/xtcinca/mylib/perl
```

3.11 Speaker Information File

A speaker information file has one line per speaker. Each line must begin with a speaker's ID. Remaining contents may be customized in general. However, most scripts assume, that the first four tokens on each line contain the speaker's ID, group membership, age and score. The number and meanings of the remaining, extra tokens on each line is arbitrary, but should be uniform for each speaker entry.

```
<SpkID 1> <Group> <Age> <Score> <Extra1> <Extra2> ...
<SpkID 2> <Group> <Age> <Score> <Extra1> <Extra2> ...
...
```

`<Group>` is either the group ID, e.g., G1, the cluster, e.g., C3, or the first language of the speaker, e.g., J (= Japanese). See `spkdata.rb` for information on language symbols used. A speaker information file for known speakers can be generated, when passing a list of non-native speaker IDs registered in library `spkdata.rb` through script `spkinfo.rb`.

Chapter 4

Detailed results

This chapter will give detailed information about experimental results. Scripts and script fragments to reproduce some of the results together with important parameters are printed.

4.1 Data sets

Information about the training, validation and test data sets is shown in table 4.1. The number of words in each data set is determined after splitting compound words like "HOW_MAY_I_HELP_YOU" into single word sequences, i.e. "HOW", "MAY", "I", "HELP", "YOU", and splitting words with apostrophes like "I'M", into "I" and "M". However, the following words were not split: "YOU'RE", "WOULDN'T", "WON'T", "WASN'T", "TRAVELER'S", "SHOULDN'T", "O'CLOCK", "MA'AM", "LORI'S", "LET'S", "ISN'T", "THAT'S", "IT'S", "HAVEN'T", "HASN'T", "DON'T", "DOESN'T", "DIDN'T", "COULDN'T" and "CAN'T". The set "train" was used for training of AMs from scratch or adaptation of a speaker-independent baseline system. Only the set "test" was used in evaluation experiments. The set "vali" was employed in order to adjust decoder parameters, e.g., LM scale. Data partitioning different from that shown in table 4.1 was not done.

Table 4.1: *Training, validation and test data sets for each speaker with contents, ID of utterance sets, number of words and sentences, and average duration in seconds.*

Data Set	Utterance Set	Contents	# Words	# Utterances	Duration (s)
Train	SX	48 phonetically rich sentences from the TIMIT database	401	48	min: 185 avg: 240 max: 402
Train	TAC22012 TAS12008 TAS12010	Hotel reservation dialogs (HotelDialog)	252 104 144	19 9 12	min: 192 avg: 244 max: 322
Vali	demo02	HotelDialog	70	10	avg: 40
Test	TAS22001 TAS32002	HotelDialog	162 182	10 13	min: 129 max: 243

The speech data sets were almost completely available for 98 non-native speakers, whose IDs are: F002, F004, F009-F014, F018-F026, M001, M006, M010-M016, M021-M093.

However, for experiments with non-natives only the data of the following male speakers was utilized: M001, M006, M010-M016, M021-M093.

For decoding experiments with the SI native baseline, the test data of the following six native speakers was used: M002, M005, M048, M094, M095 and M096. Adaptation of the SI native baseline to obtain SD models was done for the following six native speakers: F002, F003, F004, F007, M005 and M048. These SD models were evaluated with the test data of the same six native speakers.

4.2 Language model

The language model was built with HTK LM tools. 235 dialogs from ITL Hotel reservation tasks comprising 6,460 utterances with 66k words were used to estimate a Good-Turing discounted bigram. No dialog transcripts of the test data were included in the training data. The dictionary contains 8,875 pronunciation variants of 7,311 words. The OOV rate for the test set is 1.0, the perplexity 32.

```
Experiment directory: language_model
```

4.3 Native SI AM experiments

In the beginning, a native AM with 37,413 utterances of the LDC WSJ corpus was trained. A monophone and a biphone AM were trained with scripts {1,2}. Furthermore, a crossword-triphone AM, which was built by K.Markov with HTK with the same data, was used. The HMMs of all AMs had three states.

The beamsize for all recognition experiments was set to 200.0. The LM scale factor was determined with the validation data set. Figure 4.1 shows the word accuracy vs. LM scale plot when decoding the utterances of the validation data set. While performance for native speakers was uniform regardless of the LM scale factor, the performance for non-native speakers with rather low word accuracy increased with a higher scale factor. Since the performance for non-native speakers with rather high word accuracy decreased for larger scale factors, the scale factor was set to 16.0. This setting is considered as a fair trade-off for all non-native speaker groups.

The recognition accuracy for the test data of each non-native speaker group with the SI native models is shown in Tables 4.2, 4.3 and 4.4. For each speaker group, the number of speakers (#spk), the mean recognition rate (mean) together with its maximum (*max*), minimum (*min*) and standard deviation (stddev) is given.

```
Experiment directories:
  baseline_mono, baseline_biph,
  adapted_mono, adapted_biph, adapted_triph
```

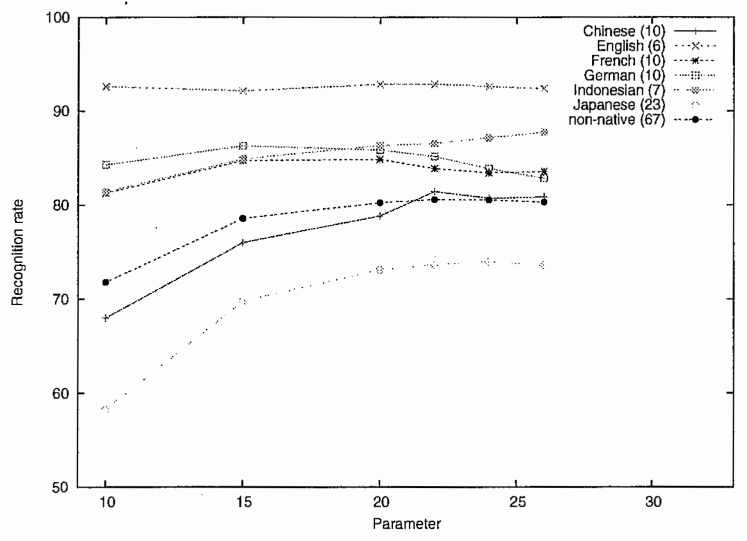

Figure 4.1: *LM scale vs. performance for decoding with monophone SI baseline AM.*

WA	mean	stddev	*min*	*max*	#spk
English	82.4	5.3	76.5	89.5	6
German	75.7	6.5	62.5	86.0	15
French	71.9	8.3	50.0	80.8	15
Indonesian	70.7	5.0	61.6	82.3	15
Japanese	55.4	16.2	25.0	82.6	15
Chinese	63.3	10.0	43.9	79.7	15

Table 4.2: *WA with monophone SI native AM. Mix = 16, Beam = 200.0, LM scale = 16.0*

WA	mean	stddev	*min*	*max*	#spk
English	82.5	5.0	75.0	88.4	6
German	72.5	7.2	57.7	81.7	15
French	66.9	8.2	51.6	75.6	15
Indonesian	63.1	9.9	48.0	81.9	15
Japanese	42.9	18.8	10.0	72.4	15
Chinese	54.9	12.0	36.3	75.9	15

Table 4.3: *WA with biphone SI native AM. Mix = 10, Beam = 200.0, LM scale = 16.0*

WA	mean	stddev	*min*	*max*	#spk
English	85.5	7.2	70.6	93.0	6
German	69.6	11.3	46.5	84.6	15
French	62.1	10.0	43.3	76.5	15
Indonesian	51.7	13.4	23.3	80.5	15
Japanese	31.4	21.9	1.2	65.4	15
Chinese	39.0	19.3	13.9	73.8	15

Table 4.4: *WA with triphone SI native AM. Mix = 12, Beam = 200.0, LM scale = 15.0*

4.4 Native GD AM experiment (baseline)

Since only male non-native speakers are utilized in further experiments, a gender-dependent (GD) native baseline AM was constructed for a fairer comparison of the baseline system with any of the proposed approaches. The GD model was obtained by MAP adaptation of the SI model with the speech data of all male speakers in the LDC WSJ corpus. The MAP adaptation parameter was set to 10.0.

Table 4.5: *WA with monophone GD native AM. Mix = 16, Beam = 200.0, LM scale = 16.0*

WA	mean	stddev	*min*	*max*	#spk
English	81.5	6.5	73.5	88.4	6
German	75.7	5.6	65.4	84.0	15
French	73.8	6.3	57.3	81.4	15
Indonesian	71.9	5.2	63.1	86.0	15
Japanese	55.6	16.1	28.8	82.6	15
Chinese	63.7	10.4	46.2	80.2	15

`Experiment directories: baseline_mono_male, adapted_mono`

4.5 Speaker-dependent AM experiment

A reference value for maximum possible performance can be obtained by decoding with speaker-dependent models. The SD model for each speaker was constructed by adaptation of the SI monophone AM with the "train" data set. The MAP adaptation parameter was set to 1.0, since in preliminary decoding experiments with the "vali" set it was found out, that the performance descreased for higher values.

Table 4.6: *WA with monophone SD AM. Mix = 16, Beam = 200.0, LM scale = 16.0*

WA	mean	stddev	*min*	*max*	#spk
English	92.6	2.4	88.7	95.3	6
German	87.2	4.2	79.7	93.6	15
French	87.5	2.6	82.6	91.3	15
Indonesian	87.7	2.8	82.8	93.6	15
Japanese	84.6	5.7	74.1	93.6	15
Chinese	82.8	5.3	72.1	90.7	15

`Experiment directory: adapted_mono`

4.6 Speaker clustering

The following sections explain the steps necessary to do clustering in Eigenspace together with script fragments.

Step 1: Construction of SD models. The single-mixture SI monophone AM trained with all the data of LDC WSJ is adapted with the training data of each non-native speaker in order to obtain one SD model for each speaker. MAP adaptation with adaptation parameter $\tau = 1.0$ was applied.

```
mapscf    = 1.0
mixes     = 1
regcls    = 32
statfile  = "basemodels/stats.mono.mix#{mixes}"
modelfile = "basemodels/hmm.mono.mix#{mixes}"
phonelist = "basemodels/monophones"
for spk in $spklist
  targetdir  = "speakers/#{spk}"
  targetfile = "hmm.mono.mix#{mixes}.sd.map#{mapscf}"
  labelfile  = "#{spk}.adapt.aligned.phone.mlf"
  if File::exists? ("#{targetdir}/#{labelfile}")
    and not File::exists? ("#{targetdir}/#{targetfile}") then
    if not File::exists? (targetdir) then
      my_system ("mkdir #{targetdir}")
    end
    my_system ("htk-adapt.rb -a lists/#{spk}.adapt.mfcc -h #{modelfile}
                -d speakers/#{spk} -i adapt -l #{targetdir}/#{labelfile}
                -p #{phonelist} -s #{statfile} -r #{regcls} -j #{mapscf}
                -d #{targetdir} -o #{targetfile}")
  end
end
```

Step 2: Eigenspace computation. In the foreach loop of the following script, the model parameters, i.e. mean vector, covariance matrix, etc. are extracted from each SD model. After that, script `eigenvoice.py` computes the basis of the Eigenspace, projects each model into the Eigenspace and writes the representative vector of all speakers to a single file. Furthermore, files with Eigenvalues, Eigenvectors, Eigenvalue ratio and accumulated Eigenvalue ratio are written.

```
set spklst = ‘cat spklist | stripnl2.rb‘
mkdir sdmodelparams
foreach spk (${spklst})
  htk-extractvoices.rb -h speakers/${spk}/hmm.mono.mix1.sd.map1
       | modasc2bin.py > sdmodelparams/${spk}_all.params
end
mkdir eigenparams
eigenvoices.py -s sdmodelparams -t eigenparams -i spklist -p all
```

Step3: Clustering. The following script shows an example for how to do clustering with all implemented clustering algorithms. The last call to `clusterspk.py` does k-means clustering with knowledge-based initialization. `clusterdata.*` are speaker information files, which contains the cluster index for each speaker. The cluster configuration in that file will be used later in section 4.8 for building cluster-dependent acoustic models.

```
set algos    = (k-means h-min h-max h-mean h-avg)
set basename = clusterdata
set minfeat  = 1
set maxfeat  = 20
set spkinfo  = "spkinfo"
set emodels  = "eigenparams/eigenmodels_all"
set k        = 5
set iters    = 10
foreach algo ($algos)
  set cdata = $basename.cl_$k.$algo.$minfeat-$maxfeat
  clusterspk.py -a $algo -c $k -m $emodels -s $spkinfo
                -f $minfeat,$maxfeat-g $cdata -i $iters
end
set cdata = clusterdata.cl_$k.k-means.$minfeat-$maxfeat.kb_init
clusterspk.py -g $cdata -a k-means -c $k -m $emodels
              -s $spkinfo -f $minfeat,$maxfeat -i $iters -k
```

4.6.1 Examples of cluster configurations

In this section, the results of speaker clustering in Eigenspace with different parameters and algorithms are summarized. Table 4.7 shows the result of clustering 74 non-native speakers[1]. r_k is the accumulated Eigenvalue ratio, if only the first k Eigenvectors belonging to the k largest Eigenvalues of a maximal n-dimensional Eigenspace are used. λ_i is the i-th largest Eigenvalue.

Among the hierarchical clustering algorithms, only "h-max" produced balanced clusters, even if the dimension of the Eigenspace was large. The k-means algorithm produced balanced clusters regardless of the Eigenspace's dimension. The cluster configuration generated by "k-means" with knowledge-based initialization was utilized for building cluster-dependent acoustic models.

```
Experiment directory: sd_mono_mix1_map
```

[1]Indonesian native M087 learned English in Hungary, so he was excluded during clustering

Algorithm	MinDim	MaxDim	$r_k = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{n} \lambda_i}$	Clusters	Distribution				
k-means (random init)	1	1	0.788	5	16	10	21	9	18
		2	0.817		16	20	8	20	10
		3	0.844		7	14	8	27	18
		4	0.860		4	20	11	15	24
		5	0.871		6	15	6	31	16
		10	0.907		6	17	6	28	17
		20	0.941		5	19	4	21	25
k-means (know-ledge based init)	1	1	dto.	5	16	18	13	9	18
		2			18	11	17	13	15
		3			20	18	15	6	15
		4			18	22	10	12	12
		5			13	21	13	13	14
		10			10	21	11	11	21
		20			10	19	15	13	17
h-min	1	1	dto.	5	66	3	2	1	2
		2			70	1	1	1	1
		3			70	1	1	1	1
		4			70	1	1	1	1
		5			69	2	1	1	1
		10			69	1	2	1	1
		20			1	70	1	1	1
h-max	1	1	dto.	5	20	12	13	24	5
		2			15	8	31	10	10
		3			39	10	14	9	2
		4			4	17	14	33	6
		5			5	23	28	14	4
		10			16	21	29	5	3
		20			5	13	29	23	4
h-avg	1	1	dto.	5	15	36	13	9	1
		2			25	8	17	22	2
		3			56	8	3	4	3
		4			4	12	43	12	3
		5			4	58	7	2	3
		10			5	63	3	2	1
		20			4	64	4	1	1
h-mean	1	1	dto.	5	15	36	13	9	1
		2			24	8	34	6	2
		3			46	14	8	4	2
		4			4	15	1	51	3
		5			64	1	2	3	4
		10			69	1	2	1	1
		20			4	66	2	1	1

Table 4.7: *Distribution of non-native speakers in each cluster for each clustering algorithm.*

4.7 Accent-dependent models

In the beginning, three methods for building accent-dependent acoustic models were examined:

1. MAP adaptation of the native SI monophone AM described in section 4.3
2. training of AMs from scratch
3. training of AMs with speech data which was pre-segmented by a forced-alignment with the native SI monophone AM

The recognition performance with these three model types is summarized in Tables 4.8, 4.9 and 4.10. No cross-validation was carried out for these three decoding experiments. The models built with method (1) had 16 mixtures, models built with methods (2) and (3) 10 mixtures. The complexity of the remaining model characteristics (three states, diagonal covariance matrix) was uniform.

The results show, that models built by methods (2) and (3) perform better (higher mean WA, smaller standard deviation of WA for most accent groups) than models built by method (1). Performance for models built with method (2) and (3) is almost equal. Despite the difference is not significant, models built with method (3) may perform better in reality. Consequently, method (3) was employed for AM construction in further experiments.

Table 4.8: *Results for AD AMs built by MAP adaptation of SI monophone baseline AM (1).*

WA	mean	stddev	*min*	*max*	#spk
German	83.0	5.0	74.4	91.3	15
French	83.3	4.2	72.4	88.4	15
Indonesian	84.5	3.6	79.7	92.2	14
Japanese	78.2	7.0	63.7	91.0	15
Chinese	76.9	6.0	63.7	88.4	15

Table 4.9: *Results for AD AMs built by training from scratch (2).*

WA	mean	stddev	*min*	*max*	#spk
German	85.0	5.1	73.3	92.2	15
French	86.2	2.9	81.1	92.2	15
Indonesian	87.3	3.9	80.5	94.5	14
Japanese	83.9	3.9	78.4	90.7	15
Chinese	80.9	4.7	73.3	88.7	15

Table 4.11 shows the result for AMs built with method (3) with cross-validation.[2] The number of mixtures is 10. Recognition performance with accent-dependent models is significantly better than with the native GD baseline AM for all non-native speaker groups.

[2]Speaker M087 is included in the Indonesian speaker group only for evaluation

Table 4.10: *Results for AD AMs built by bootstrapping with pre-segmented speech data obtained by force-alignment with native SI monophone baseline AM (3).*

WA	mean	stddev	*min*	*max*	#spk
German	85.4	4.4	77.9	93.6	15
French	86.7	3.5	79.9	93.0	15
Indonesian	86.9	3.0	82.6	94.5	14
Japanese	84.8	3.2	79.1	90.7	15
Chinese	81.3	3.6	75.9	87.2	15

Table 4.11: *Results for AD AMs built with method (3) and leave-one-speaker-out cross-validation.*

WA	mean	stddev	*min*	*max*	#spk
German	82.7	5.3	72.1	91.3	15
French	84.4	3.7	77.9	91.3	15
Indonesian	85.4	4.2	77.0	92.2	15
Japanese	82.2	3.8	74.4	89.0	15
Chinese	77.3	5.6	66.3	88.4	15

```
Experiment directories:
 ad_mono_adapt_map
 ad_mono_train_scratch
 ad_mono_train_sialign
```

4.8 Cluster-dependent models

Decoding with cluster-dependent models was also significantly better than with the native GD baseline AM [3]. The number of mixtures is 10. However, the performance was slightly lower than with accent-dependent models, except the Chinese speaker group. This phenomenon can be explained by the fact, that the speakers in the non-native database were from several areas of mainland China, also including speakers from Hong Kong and Taiwan.

Table 4.12: *Results for CD AMs built with method (3) and leave-one-speaker-out cross-validation.*

WA	mean	stddev	*min*	*max*	#spk
German	80.1	5.6	68.3	88.7	15
French	82.8	3.6	76.5	86.6	15
Indonesian	82.9	4.6	76.5	90.7	15
Japanese	82.1	3.3	75.4	87.5	15
Chinese	75.5	6.3	62.8	85.8	15

[3]Speaker M087 is included in cluster C4 only for evaluation to make comparisons easy

```
Experiment directory: cd_mono_train_sialign
```

4.9 Non-native models

Recognition results with a speaker-independent accent-independent non-native monophone acoustic model are shown in Table 4.13. 3-fold cross-validation was carried out, i.e. 50 speakers for training and 25 for evaluation. To get a model which can cope with non-native speech of all five accent groups, the speech data of ten speakers of each group was included in the training data set. The same was done for the test data, i.e. five speakers from each accent group. The number of mixtures per state was increased until 16.

Table 4.13: *Results for NN AMs built with method (3) and 3-fold cross-validation.*

WA	mean	stddev	*min*	*max*	#spk
German	80.5	6.7	67.7	90.4	15
French	83.1	4.3	72.1	89.0	15
Indonesian	83.2	2.9	77.0	87.2	15
Japanese	79.7	4.6	72.4	87.8	15
Chinese	79.8	4.7	69.5	88.1	15

```
Experiment directory: nn_mono_train_sialign
```

4.10 Parallel decoding

In the following, two script fragments are printed. The first is for doing parallel decoding with all accent-dependent models. The second is for AM score extraction and selection of the hypothesis with maximum AM score. The selection can also be based on the combined AM plus LM score, but selection by considering only the acoustic likelihood was more promising. The script for parallel decoding with cluster-dependent models is the same but with different paths and group symbols instead of language symbols.

```
#
# Script for parallel decoding
#

lexikon     = "lexikon/hotel_8k_sp_sil.lex"
wordnet     = "lm/hotel_big.wordnet"
phonelist   = "monophones"
targetdir   = "recout_lm8k_AD"
mixes       = 10
lmweight    = 16.0
beam        = 200.0
```

```
#
# decoding with each AD model
#
for spk in spklist
  (natlang, age, score) = getspkinfo (spk)
  for lang in ["C","J","I","G","F"]
    if lang == natlang then
      model = "speakers_AD/#{spk}/models.#{lang}.4.mix#{mixes}/hmm.trained"
    else
      model = "ad_models/hmm.mono.mix#{mixes}.#{lang}"
    end
    flist = "lists/#{spk}.eval.mfcc"
    targetfile = "#{targetdir}/#{lang}/#{spk}.eval.recout.mlf"
    stdoutlog  = "#{targetdir}/#{lang}/#{spk}.eval.stdout"
    my_system ("HVite -i #{targetfile} -H #{model} -S #{flist} -p 0.0
                  -s #{lmweight} -t #{beam} -w #{wordnet}
                  #{lexikon} #{phonelist} | tee #{stdoutlog}")
  end
end

#
# Script for hypothesis selection
#

targetdir  = "recout_lm8k_AD"
scorebase  = "scores_AD"
resultbase = "result_AD"
flistbase  = "flist_AD"
htkresult  = "result_AD_parallel"
reflabels  = "labels/all.mlf"

#
# extract acoustic scores
#
for spk in spklist
  (natlang, age, score) = getspkinfo (spk)
  scorefile  = scorebase+"."+natlang
  list = []
  for lang in ["C","J","I","G","F"]
    targetfile = "#{targetdir}/#{lang}/#{spk}.eval.recout.mlf"
    stdoutlog  = "#{targetdir}/#{lang}/#{spk}.eval.stdout"
    my_system ("cat #{stdoutlog} | htk-vitestdoutparse.rb
                         -c > #{tmpfile}.#{lang}")
    list << "#{tmpfile}.#{lang}"
  end
  my_system ("paste "+list.join(" ")+" >> #{scorefile}")
  my_system ("rm -f "+list.join(" "))
  my_system ("cat lists/#{spk}.eval.mfcc >> #{flistbase}.#{natlang}")
end

#
# hypothesis selection
#

# implicit argmax (in script recrate.pl)
my_system ("recrate.pl -l #{langstr} -s #{scorebase} -r #{resultbase}")

# read selected hypothesis from MLF files and write them to a new MLF file
for spk in spklist
  next if spk[0..0] == "F"
  (natlang, age, score) = getspkinfo (spk)
  next if not langs.member? (natlang)
  my_system ("paste #{flistbase}.#{natlang} #{resultbase}.#{natlang}
              | awk '{ print $1, $2; }' | grep #{spk} > #{tmpfile}")
  mlf = {}
  for lang in ["C","J","I","G","F"]
    sourcefile = "#{targetdir}/#{lang}/#{spk}.eval.recout.mlf"
    mlf[lang] = MLF.new (sourcefile, "TWL")
    mlf[lang].read (nil,"rec")
  end
  targetfile = "#{targetdir}/#{spk}.eval.recout.mlf"
  targetmlf = MLF.new (targetfile, "TWL")
  fd = open (tmpfile)
  while line = fd.gets
    tokens = line.chomp.split (" ")
    ident  = tokens[0].sub(/features\//,"").sub(/\.mfcc/,"")
    index  = tokens[1].to_i - 1
    targetmlf.set_data (ident, mlf[langs[index]].get_data(ident))
  end
  targetmlf.write ("rec")
end
```

Detailed results for the parallel decoding experiments are shown in Tables 4.14 and 4.15.

```
Experiment directory: parallel_decoding_ALL
```

Table 4.14: *Results for parallel decoding with AD models and leave-one-speaker-out cross-validation.*

	mean	stddev	*min*	*max*	#spk
German	80.6	7.1	63.7	90.4	15
French	83.6	3.5	78.2	90.7	15
Indonesian	83.0	7.4	65.4	90.7	15
Japanese	80.4	5.0	71.8	89.0	15
Chinese	75.5	6.2	61.3	87.5	15

Table 4.15: *Results for parallel decoding with CD models and leave-one-speaker-out cross-validation.*

	mean	stddev	*min*	*max*	#spk
German	80.1	5.6	68.3	88.7	15
French	82.8	3.6	76.5	86.6	15
Indonesian	82.9	4.5	76.5	90.7	15
Japanese	82.0	3.6	73.3	87.5	15
Chinese	75.5	6.3	62.8	85.8	15

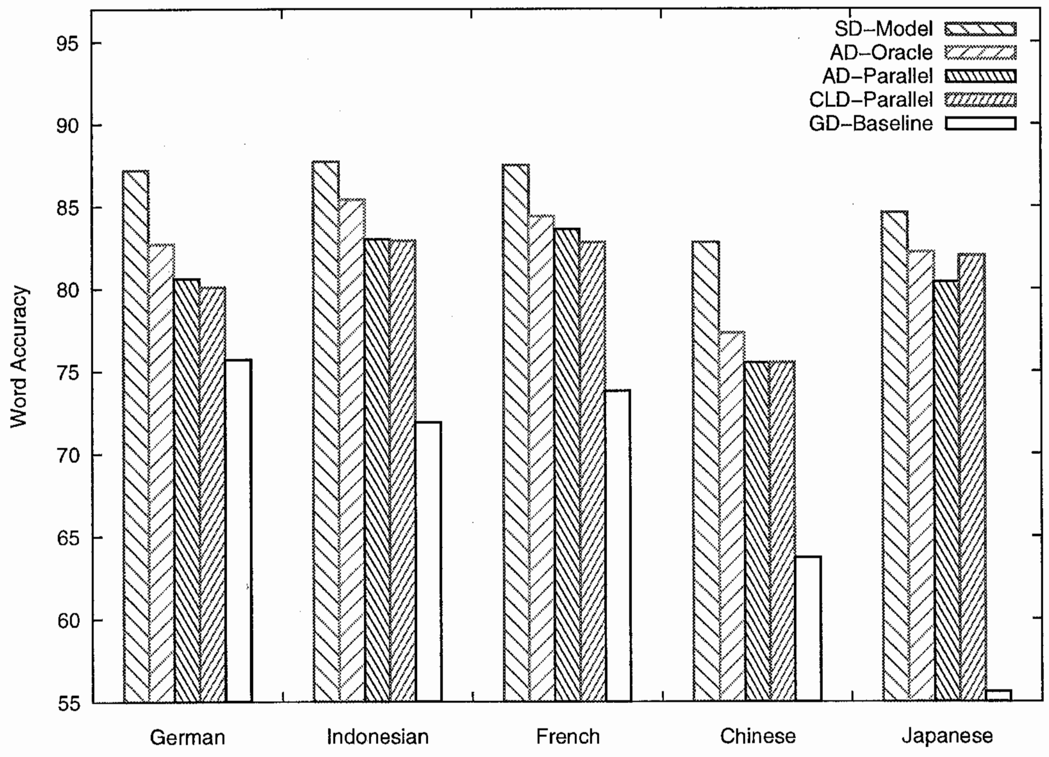

Figure 4.2: *Summary of experimental results.*

Chapter 5

Survey of AM adaptation methods

This chapter gives a survey of techniques for model-based speaker adaptation. Frequently applied adaptation methods like MAP, MLLR and Eigenvoice are described. Since the focus of this thesis are subspace-based adaptation schemes, eigenspace-based adaptation methods and adaptation methods employing speaker clustering are covered in more detail. Finally, some important results from publications on non-native speech recognition are cited.

For the basic theory of Hidden Markov Models (HMM), especially how HMM parameters can be estimated iteratively with the Baum-Welch Algorithm, which belongs to the class of Expectation-Maximization (EM) algorithms, see for example [39].

5.1 Introduction

It is well known, that performance of an automatic speech recognition (ASR) system drops substantially, if there is a mismatch between training data and test data set conditions. Speaker-dependent (SD) systems perform better than speaker-independent (SI) systems if speech of the speaker the system was trained on is to be recognized. In telephone-based or terminal-based applications for speech recognition, i.e. dialog systems, the recognizer should work well for every speaker who is expected to encounter the system. Usually there are several groups of potential users with each group having different speech characteristics, comprising aspects like gender, age, dialect or foreign accent. The result is poor recognition performance for a speaker group which was not seen during the training phase of the recognizer. One possible approach would be to collect enough data for each relevant speaker group, build one recognizer from that data for each group and build a classifier which can differentiate among groups to select the most suitable recognizer for incoming speech data. However, this approach is not feasible in practice, since there are too many possible speaker groups to be covered and building reliable classificators to identify the correct speaker group is difficult.

Popular approaches to cope with inter-speaker variability are techniques for speaker adaptation. There exist feature-based and model-based acoustic adaptation techniques. In feature-based adaptation, also called normalization, only the speech features are modified to alleviate variability. One example is Vocal Tract Length Normalization (VTLN) for reducing effects on speech sound introduced by gender (male vs. female speakers) or age (children vs. adults) of speakers. However, normalization techniques alone do not suffice to model inter-speaker variability due to foreign ac-

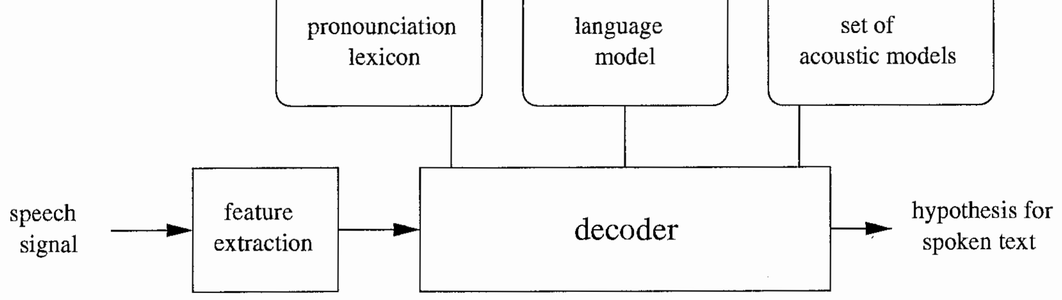

Figure 5.1: General architecture of a speech recognizer for LVCSR

cent or dialect. Beside doing pronunciation modeling, i.e. finding appropriate phone strings for each word, many researches propose methods for adaptation of the acoustic model parameters of the speech recognizer. State of the art speech recognition systems use Hidden Markov Models (HMMs) for acoustic modeling. In this context adaptation means that all or some parameters of the HMMs (e.g. parameters of mixture distributions or HMM probabilities) are altered so that recognition performance improves for a certain speaker or speaker group.

The most cited methods for adaptation of acoustic models are Maximum a posteriori (MAP) adaptation, Maximum Likelihood Linear Regression (MLLR) and Eigenvoice. If there is only little adaptation data available, robust adaptation by MAP or MLLR is impossible, because the number of parameters to be estimated is too large. In contrast to MAP and MLLR, the Eigenvoice approach improves recognition performance even if there are only a few seconds of adaptation data available.

Similar to the Eigenvoice approach are adaptation methods, which cluster training speakers into groups with common characteristics and then selecting the speaker group or the speakers, which are acoustically closest to the speaker the acoustic model has to be adapted to. Finally, recognition is done with the acoustic model derived from the selected speakers.

5.2 General remarks on adaptation

The general architecture of a speech recognizer for large vocabulary continuous speech recognition (LVCSR) is shown in figure 5.1. The main parts are the pronunciation dictionary, which defines the mapping of words to subwords units, the language model by which the probability of a word sequence can be calculated, the acoustic models for the subword units, usually represented by a large set of HMMs, and the decoder, which combines the information of the three former modules to extract a hypothesis for the spoken text from the feature vector sequence of the speech signal.

There are several important aspects of adaptation in general by which speaker adaptation techniques can be classified. These are summarized in table 5.2, since they frequently appear in publications on speaker adaptation. Feature-based adaptation accomplish adaptation only by transformations of the feature space. These adaptation methods will not be described further, since this thesis concentrates on model-based adaptation techniques.

The starting point for model-based adaptation is usually a well-trained speaker-independent (SI) acoustic model. In the focus of model-based adaptation techniques are the parameters of

Aspect	Meaning	Example
Feature-based	Adaptation by transforming the feature space	VTLN [31]
Model-based	Adaptation by transforming acoustic model parameters Selection or interpolation of the best fitting models	MLLR [33]
Supervised	Transcription of adaptation data is known	
Unsupervised	Transcription of adaptation data is *un*known Adaptation data must be decoded before adaptation	
Static, Offline	Adaptation data is available a priori Adaptation is done in advance only once	*dictation system*
Dynamic, Online	Adaptation data is obtained at run time Adaptation is done incrementally	*dialog system*
Rapid	Only a small amount of adaptation is available Low computational cost of adaptation process	Eigenvoice [28]
Structural	Organization of parameters in a hierarchical tree Control adaptation by amount of adaptation data	SMAP [40]

Table 5.1: Classification of adaptation methods

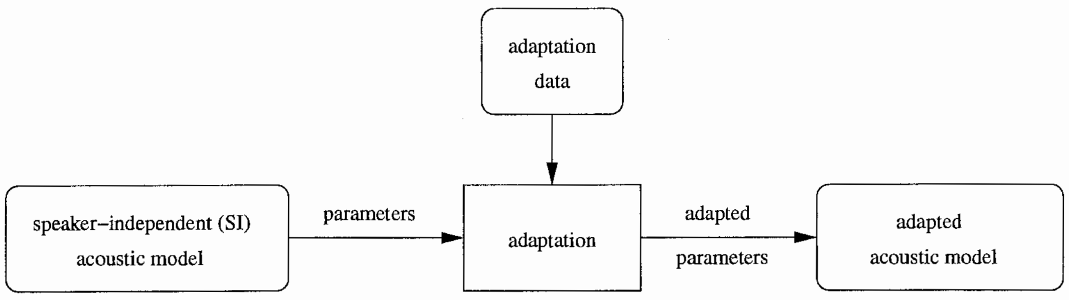

Figure 5.2: Outline of speaker adaptation

the acoustic models. These are usually the initial state, state transition and state occupancy probabilities of the HMMs, together with mixture weighting coefficients, mean vectors and co-variance matrices of Gaussian mixtures for the output probability densities of each HMM state. For reliable estimation of the parameters tens of hours of speech data are necessary. Provided a certain amount of adaptation speech data, the task is to modify the parameters in order to enhance the performance of the speech recognizer on test data, which has similar characteristics as the adaptation data. Figure 5.2 briefly sketches the idea of model-based speaker adaptation.

The choice of the appropriate adaptation method depends critically on the amount of adaptation data available and for which purpose adaptation has to be applied. If a dictation system has to be adapted to a specific user, it is possible to provide much adaptation data and to adapt the system even it takes some resources, especially computational power and users time, because adaptation can be carried out offline. But if a telephone-based dialog system has to be adapted online to the current caller, it is not reasonable to let the user wait for a long time or ask him to provide large amounts of adaptation data.

The following section briefly dwells on MAP and MLLR, two adaption methods which can only be applied effectively, if a large amount of adaptation data is available. The Eigenvoice approach

Author	Task/DB	Technique	Outline of approach
[22]	RM	MAP	$\hat{\vec{\mu}} = \vec{\mu} + \vec{b}$
[33]	WSJ	MLLR	$\hat{\vec{\mu}} = \vec{A}\vec{\mu} + \vec{b}$
[29]	Isolet	Eigenvoice	$\hat{\vec{\mu}} = \vec{\mu} + \sum_{i=1}^{K} a_i \vec{e}_i$
[24]	RM	SCW,RSW	Weighting of speaker- or cluster-dependent models
[11]	WSJ	PCA,MLLR	Constrain MLLR transformations to eigenspace
[3]	WSJ	SAT	Adaptation of speakers before model training
[21]	Dictation	CAT	Weighting of partitions of cluster means

Table 5.2: List of important publications regarding acoustic model adaptation

which addresses the problem of rapid speaker adaptation will be introduced in section 5.4. A short overview to important publications regarding speaker adaptation methods of acoustic model parameters is assembled in table 5.2.

5.3 Methods for large amounts of adaptation data

5.3.1 Maximum a posteriori (MAP)

When estimating unknown parameters of a probability density function an optimization criterion has to be defined first. Frequently applied estimation techniques are (ML) estimation and maximum a posteriori (MAP) estimation. The latter is also called Bayesian estimation.

ML estimation tries to maximize the likelihood P of a training data set $\mathcal{X}$. If $\vec{\Theta}$ is the parameter of a tuple of parameters to be estimated, the optimization criterion of ML estimation can be expressed as follows:

$$\vec{\Theta^*} = \text{argmax}_{\vec{\Theta}} P(\mathcal{X}|\vec{\Theta}) \tag{5.1}$$

In MAP estimation prior densities for the unknown parameters $\vec{\Theta}$ are assumed. The optimization criterion is then altered to

$$\vec{\Theta^*} = \text{argmax}_{\vec{\Theta}} P(\mathcal{X}|\vec{\Theta})P(\vec{\Theta}) = \text{argmax}_{\vec{\Theta}} P(\vec{\Theta}|\mathcal{X}) \tag{5.2}$$

With the Baum-Welch Algorithm ML estimates for HMM parameters are obtained. In [22] a scheme for MAP estimation of HMM parameters is derived. The authors assume the Dirichlet density for mixture weights and HMM probabilities (initial, state occupancy and state transition). Normal-Wishart probability density function is assumed as prior for means and covariances of Gaussian mixture components.

The MAP estimation scheme for HMM parameters can be used if not enough training data for reliable ML estimation of HMM parameters is available. The importance of MAP for this thesis is its applicability to speaker adaptation.

For speaker adaptation the parameters of robustly trained HMMs are employed as a priori information. First, the adaptation data is decoded with the existing model in order to assign each feature vector to mixture density components and to obtain statistics on state transition and state occupancies. The assigned data and statistics are used for reestimation of HMM parameters. The MAP reestimation formulas are very similar to Baum-Welch estimation formulas except that each formula consists of a weighted sum of prior information and adaptation data. Not all HMM parameters have to be adapted. Most effective is an adaptation of means of Gaussian mixture densities. The reestimation formula for the mean vector of a mixture density of an HMM is as follows:

$$\hat{\vec{\mu}}_k = \frac{\tau_k \vec{\mu}_k + \sum_{t=1}^{T} c_{kt} \vec{y}_t}{\tau_k + \sum_{t=1}^{T} c_{kt}} \tag{5.3}$$

$\vec{\mu}_k$ is the prior value of the mean vector of the k-th mixture component, $\vec{y}_t$ represents the adaptation data and c_{kt} is the posterior probability of y_t being produced by the k-th mixture component. τ_k weights prior information against adaptation data. If $\tau_k = 0$ the prior information can be canceled and the formula reduces to the conventional ML estimate for mean vector $\vec{\mu}_k$ in Baum-Welch formulas.

To alleviate the disadvantage of MAP, that it can only be applied successfully if much adaptation data is available, a structural version of the MAP adaptation scheme was proposed in [40].

Structural adaptation methods organize the acoustic model parameters in a hierarchical tree. The amount of adaptation data available controls at which nodes at which tree levels adaptation should be carried out first. Adaptation of a parent node affects adaptation of its child nodes. For example, if there is much adaptation data, which can be associated with certain leaf nodes, parameters of these leaf nodes can be adapted independently from others. Otherwise, parameter adaptation (e.g. transformation or shift of Gaussian mean vector) of the corresponding parent node is adopted.

Some evaluation results for MAP adaptation can be found in [22] [49] [43] [51]. Results presented in [51] show that adaptation of means is more effective than adaptation of covariances and adaptation of covariances is more effective than adaptation of mixture weights.

5.3.2 Linear Regression (LR)

An approach to speaker adaptation using linear regression was first proposed in [32] by Leggetter et al. and later improved in [33].

The idea is to transform the mean vectors of the Gaussian mixture components by a linear transformation.

$$\hat{\vec{\mu}}_k = \vec{A}_k \vec{\mu}_k + \vec{b}_k \tag{5.4}$$

Index k indicates, that each mean vector is transformed individually. In [32] it is mentioned that this would lead to a complete reestimation of the all acoustic models. On the other hand, using only one global transformation for all mean vectors would lead to too poor adaptation results as more adaptation data becomes available.

To make appropriate use of the adaptation data available, Leggetter et al. employed the idea of grouping mixtures or mixture components into regression classes, which are similar in terms of a distance measure. Depending on the amount of adaptation data available the number of regression classes and the grouping of mean vectors is determined and one linear transformation for each regression class is obtained. The parameters of the linear transformation $\vec{A}$ and $\vec{b}$ are estimated by the maximum likelihood (ML) criterion. This explains the approach is called maximum likelihood linear regression (MLLR).

Instead of maximum likelihood (ML), Bayesian estimation of the transformation parameters is also possible. This approach is named MAPLR and was presented in [9]. Like in the MAP adaptation scheme mentioned in subsection 5.3.1 a constraint for the transformation parameters is introduced by a probability density function.

In [47] estimation of transformation parameters was carried out by employing maximum mutual information (MMI) criterion. The approach was named MMILR accordingly.

5.4 Methods for sparse amounts of adaptation data

5.4.1 Eigenvoice

Inspired by the Eigenface approach for face recognition, Kuhn et al. developed a technique for rapid speaker adaptation based on Eigenvoices [28] [29].

The Eigenvoice approach can be summarized as follows: Given a large training data set with many (n) speakers and sufficient data for each speaker, the acoustic models for each speaker are constructed. The parameters of each speaker-dependent model are combined to a high-dimensional large supervector. If there are 50000 mixture components and the dimension of the feature space is 39, the dimension of the supervector is $39 * 50000 \approx 2.0 * 10^6$. Similar to MAP, adaptation can be restricted to the means of the mixture components. Principal component analysis (PCA) is applied to the set of n supervectors. The eigenvalue-ordered orthogonal vectors derived by PCA are called Eigenvoices. For adaptation it is assumed, that the parameters of the adapted acoustic models, combined in supervector $\vec{m}$, can be expressed as a linear combination of the Eigenvoices, taking only the first k principal components $\vec{e}_1, \ldots, \vec{e}_k$ of the Eigenvoice space.

$$\hat{\vec{m}} = \sum_{i=1}^{k} a_i \vec{e}_i \tag{5.5}$$

With equation 5.5 only k parameters have to be estimated for adaptation compared with at least $d^2 + d$ parameters for adaptation with a global MLLR transformation, where d is the dimension of the feature space. The justification for this procedure is that the whole space of acoustic models can be approximated by the first k Eigenvoices in the mean square error sense.

ML estimation of the weighting coefficients a_i is possible with only a few seconds of adaptation data. The estimation process was named Maximum Likelihood Eigenvoice Decomposition (MLED). Experimental results presented in [28] are promising with the limitation that only a very small scale recognition task, letters recognition, was evaluated.

In [6] the Eigenvoice approach is applied to a heterogeneous large speech database with data from 300 speakers, 100 from a proprietary database and 200 from WSJ[1], leading to significant improvements in recognition rate with only three seconds of adaptation data per speaker.

[1]Wall Street Journal

There are several recent publications on extensions and improvements of Kuhn's original approach. One drawback of the Eigenvoice approach is that it is difficult to apply to large vocabulary continuous speech recognition (LVCSR) where the number of Gaussian components is very high, because of the computational cost for PCA calculation. In [36] and [46] solutions to alleviate this problem are proposed.

Instead of applying PCA to the supervectors of all acoustic models, in [46] the supervectors were first split up into subvectors and Eigenvoices were extracted for each type of subvector. Two settings for subvector construction were suggested:

- *mixture-based:* first cluster Gaussian mixtures based on the Bhattacharyya distance, then combine parameters of each cluster to one subvector

- *feature-based:* combine acoustic model parameters of each feature group (e.g energy, MFCC, Δ MFCC, $\Delta\Delta$ MFCC) to one subvector

The proposed approach was evaluated on a Mandarin speech dictation task, leading to better results than Kuhn's original approach [28].

Similar to the mixture-based approach in [46], hierarchical Eigenvoices are proposed in [36]. Gaussian mixture components are clustered into a hierarchical tree with the k-means algorithm. Eigenvoices are calculated for the vector of means associated with each leaf node of the tree. Only the first principal components, whose sum of eigenvalues relative to the sum of all Eigenvoices is higher than a threshold, are kept to build the Eigenvoice space for each tree node. For the next higher level of the tree, model parameters are projected into each Eigenvoice subspace associated with each leaf node. The coefficients obtained by these projections are combined to one vector and again Eigenvoices are calculated for this type of vector. This procedure is continued bottom-up until the root node of the tree.

The model space to Eigenvoice space transformation can be represented by one matrix by combining projections matrices of each node of a certain tree level and then multiplying (bottom-up) these matrices together. In the adaptation phase, tree nodes with enough adaptation data associated are selected and weighting coefficients are estimated. Beside ML estimation of weighting coefficients (MLED), MAP estimation of weighting coefficients was proposed (MAPED).

An approach which combines the ideas of both MAP and Eigenvoice was presented in [7]. The Eigenvoices were incorporated as a priori knowledge into a Bayesian optimization criterion by defining an a priori probability density function with the Eigenvoices.

5.4.2 Subspace-based methods

The number of parameters in acoustic models for LVCSR is very high. To reliably estimate all these parameters for adaptation would require as much training data as is required for the complete estimation of the speaker-independent acoustic models.

The core idea of the Eigenvoice approach was to reduce the number of parameters to be estimated to make rapid adaptation possible. This was achieved by constraining the whole space of acoustic model parameters to to a very low dimensional subspace. This space was derived by applying PCA to a set of speaker-dependent acoustic model parameters concatenated to one supervector.

Instead of constraining the space of acoustic model parameters it is also possible to constrain the space of transformations of acoustic model parameters. This idea was applied to the space of MLLR transformation parameters in [8].

First the speaker-independent acoustic model is built, then MLLR is applied for each training speaker. Next, the parameters of the calculated MLLR transformations $(\vec{A}, \vec{b})$ are combined to one vector. Further steps of the procedure are analog to the Eigenvoice approach described in subsection 5.4.1.

The advantage of constraining the transformation parameter space instead of the acoustic model parameter space is a far reduced memory requirement. Even if several MLLR transformations (one transformation per regression class) are used to adapt the acoustic models for one speaker, the number of parameters of these transformations is far less than the number of parameters of the acoustic models for LVCSR.

A similar approach was presented in [11] by using principal component regression for constraining the dimension of the MLLR transformation parameter space.

5.4.3 Speaker Clustering

There are various adaptation methods in connection with the term speaker clustering. One class of such adaptation methods is similar to the Eigenvoice approach from section 5.4.1, which will be described first.

In [38] and more detailed in [24] two approaches, reference speaker weighting and speaker cluster weighting, are proposed. The parameters of the adapted model are restricted to lie in a low-dimensional subspace spanned by the acoustic models of a set of reference speakers or a set of speaker clusters. The clusters can be designed manually (e.g. clustering speakers by gender or age) or extracted automatically by agglomerating speakers with similar acoustic characteristics. Sufficient data for each each reference speaker or speaker cluster is required to reliably estimate the corresponding sets of acoustic models. In the adaptation phase, weighting coefficients of the reference models are obtained by ML criterion. The only difference to Eigenvoice approaches is, that the supervectors, which are formed by concatenation of acoustic model parameters of reference models are not orthogonal to each other.

Several other approaches carry out speaker clustering online during the adaptation phase. Starting with a set of speaker-dependent acoustic models or a set of speakers, the speakers or respectively their models, which are acoustically most similar to the adaptation data, are selected. Selection can based on the acoustic scores obtained for adaptation data during decoding [37]. Another possibility is the selection of speakers with a GMM classifier [25]. With the speech data or acoustic models of the selected speakers a new acoustic model is constructed.

An approach which tries to combine the power of speaker clustering and Eigenvoices was presented in [16]. After applying Kuhn's original Eigenvoice method [29], each speaker-dependent acoustic model is projected onto the reduced Eigenvoice space. After clustering speakers in Eigenvoice space, one acoustic model for each cluster is built. Unseen speech is decoded with the acoustic model of the cluster with the highest recognition score.

5.5 Miscellaneous adaptation methods

5.5.1 Adaptive Training

In adaptive training techniques adaptation is already employed during the training phase of the acoustic models. The idea is to build acoustic models which represent phonetic variation independent from the inter-speaker variability between speakers in the training data set.

One example is Speaker-Adaptive Training (SAT) [3], where speech data of each data speaker in the training set is normalized by an MLLR transformation in order to suppress inter-speaker variability. MLLR transformation parameters and acoustic model parameters are estimated by a joint optimization criterion.

Before recognition the acoustic models have to be adapted to incoming speech data of an unknown speaker. Again an MLLR transformation has be calculated to adjust the acoustic models to the characteristics of a new speaker. Since a larger amount of speech data is required for robust estimation of MLLR transformation parameters, the approach is not suitable for rapid adaptation.

Another adaptive training scheme is cluster adaptive training (CAT) developed by Gales [20] [21]. It is an rapid adaptation scheme and employs ideas similar to clustering and weighting in the adaptation methods based on speaker clustering and Eigenvoices.

In the approach there is a set of model clusters. Means vary between different clusters, but variances, priors and other parameters are assumed to be equal for all clusters. The Gaussian mean components are partitioned into classes. The adapted model is built by weighting the means of all model clusters. An independent weight vector is employed for each class of means.

5.5.2 Genetic Algorithms

In [30] an approach for rapid speaker adaptation based on Genetic Algorithms (GA) is proposed. Each individual of the population is represented as a string of genes, which are the means of the Gaussian mixture components. This representation is equal to the supervectors employed in the Eigenvoice approach. The fitness of an individual is defined as the time-normalized likelihood of the adaptation data of its acoustic model relative to the sum of likelihoods of all other individuals in the current population. Crossover of individual is defined as interpolation of two parents by weighting their genes with compositional coefficients, i.e. their sum is constrained to be 1.0.

5.6 Approaches to non-native speech recognition

The publication of Compernolle [10] is a literature survey that discusses the results of approaches to speech recognition of native dialects and non-native accents of a certain language. Differences between these two types of accented speech are pointed out, explaining the inherent difficulties of non-native speech recognition.

Two main categories of approaches to non-native speech recognition can be found in literature. These are:

- Lexical modeling (pronunciation modeling)
- Acoustic modeling (acoustic model adaptation)

In lexical modeling pronunciation variants are added to the pronunciation dictionary. The underlying idea is, that the pronunciation of words uttered by non-native speakers deviates from standard pronunciation. The variants can be designed manually or generated automatically from knowledge (e.g. confusion rules) which can be acquired by inspection of corrected and canonical phone labels (e.g. cf. [23]) or by consulting an expert group (e.g. cf. [41]), which defines mappings for phonemes not appearing in the first language of the non-native speaker. Experimental results of several research groups showed, that lexical modeling can improve the recognition of certain frequently appearing variants of mispronounced words to some extent [10]. However, the approach only works for systematic alterations of word pronunciations.

The two main problems with non-native speech are random pronunciation errors and the lack of a universal representative of the group of non-native speakers. The former phenomenon can be explained by the unfamiliarity of the non-native speaker with unknown words [10]. The reason for the latter is, that the effects on a second (non-native) language depend primarily on a speaker's first (native) language. For example, the results of a recent study carried out by Flege et al. [19], examining non-native English speech of Italian speakers, support the hypothesis, that the phonetic subsystems of the native and the non-native language interact through phonetic category assimilation and phonetic category dissimilation. Assimilation refers to the effect, that a speaker virtually merges speech sounds from his native language and the non-native languages, i.e. he cannot hear the differences between them and thus will not be able to reproduce the non-native sound correctly. Dissimilation means establishment of a new speech sound which has characteristics of a native language and a non-native language speech sound. In [10] the interaction of the native and the non-native language is described as the projection of pronunciation onto a lower dimensional, less discriminative space, which is defined by the intersection of the native language space and the non-native language space. Consequently, mispronunciations will vary between speakers of different nationalities. Both phenomenons, random and native language dependent pronunciation errors, restrict the applicability of pronunciation modeling.

Encouraging results for non-native English speech recognition of a single speaker group, Japanese speakers of English, were presented in [35]. Lexical modeling alone and in combination with acoustic modeling were evaluated. The authors employed a second (non-native) dictionary containing Japanese pronunciation variants of each word in the recognition vocabulary. Variants were constructed from typical Japanese accented pronunciation examples. The native and non-native dictionaries were used in parallel during recognition. Introduction of a penalty when changing dictionaries during decoding of an utterance resulted in even better performance. Additionally, adaptation of acoustic models of phonemes with MAP was also carried out, leading to further reductions of word error rate. MAP adaptation of acoustic models without pronunciation modeling leaded to significant improvements, but they were smaller than those achieved with lexical modeling.

In [44] some lexical and acoustic modeling techniques for Japanese non-native English speech recognition were compared. Improvements were achieved with pronunciation variation as well as using MLLR for acoustic model adaptation. Additional interpolation of native and non-native models led to a further increase in performance.

Further results for pronunciation modeling and acoustic model adaption for recognition of non-native speech can be found in [34] [2].

There have been approaches to non-native speech recognition with multilingual acoustic models and multilingual phoneme sets respectively. For example, the results in [48] suggest, that adding data from different native languages to the training data, a recognizer with quite uniform performance among different accents can be built. On the other hand, performance with accent specific models was higher than with multilingual models [10]. Further publications on the non-native speech recognition with multilingual acoustic models are [18] [17]. They report improvements with the multilingual approach on a small digit recognition task.

Bibliography

[1] *Handbook of the International Phonetic Association.* Cambridge University Press, 1999.

[2] Stefanie Aalburg and Harald Hoege. Approaches to foreign-accented speaker-independent speech recognition. In EUROSPEECH 2003 [15], pages 1489–1492.

[3] Tasos Anastasakos, John McDonough, Richard Schwartz, and John Makhoul. A compact model for speaker-adaptive training. In *Proceedings of the International Conference on Spoken Language Processing*, volume 2, pages 1137–1140, 1996.

[4] *IEEE Workshop on Automatic Speech Recognition and Understanding*, 2001.

[5] Nobert Binder, Rainer Gruhn, and Satoshi Nakamura. Recognition of non-native speech using dynamic phoneme lattice processing. In *Proceedings of Acoustical Society of Japan*, pages 203–204, March 2002.

[6] Henrik Botterweck. Very fast adaptation for large vocabulary continuous speech recognition using eigenvoices. In ICSLP 2000 [27], pages 354–357.

[7] Henrik Botterweck. Anisotropic MAP defined by eigenvoices for large vocabulary continuous speech recognition. In ICASSP 2001 [26], pages 353–356.

[8] Kuan–ting Chen, Wen–wei Liau, Hsin–min Wang, and Lin–shan Lee. Fast speaker adaptation using eigenspace–based maximum likelihood linear regression. In ICSLP 2000 [27].

[9] Cristina Chesta, Olivier Siohan, and Chin-Hui Lee. Maximum a posteriori linear regression for hidden markov model adaptation. In EUROSPEECH 1999 [13], pages 211–214.

[10] Dirk Van Compernolle. Recognizing speech of goats, wolves, sheep and ... non-natives. *Speech Communication*, 35:71–79, 2001.

[11] Sam-Joo Doh and Richard M. Stern. Weighted principal component MLLR for speaker adaption. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, 1999.

[12] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification.* John Wiley & Sons, Inc., 2001.

[13] *European Conference on Speech Communication and Technology*, 1999.

[14] *European Conference on Speech Communication and Technology*, 2001.

[15] *European Conference on Speech Communication and Technology*, 2003.

[16] R. Faltlhauser and G. Ruske. Robust speaker clustering in eigenspace. In ASRU 2001 [4], pages 57–60.

[17] V. Fischer, E. Janke, and S. Kunzmann. Recent progress in the decoding of non-native speech with multilingual acoustic models. In EUROSPEECH 2003 [15], pages 3105–3108.

[18] V. Fischer, E. Janke, S. Kunzmann, and T. Ross. Multilingual acoustic models for the recognition of non-native speech. In ASRU 2001 [4], pages 331–334.

[19] James E. Flege, Carlo Schirru, and Ian R. A. MacKay. Interaction between the native and second language phonetic subsystems. *Speech Communication*, 40:467–491, 2003.

[20] Mark J. F. Gales. Cluster adaptive training for speech recognition. In *Proceedings of the International Conference on Spoken Language Processing*, pages 1783–1786, 1998.

[21] Mark J. F. Gales. Cluster adaptive training of hidden markov models. In *IEEE Transactions on Speech and Audio Processing*, volume 8, pages 417–428, 2000.

[22] Jean-Luc Gauvain and Chin–Hui Lee. Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains. In *IEEE Transactions on Speech and Audio Processing*, volume 2, pages 291–298, 1994.

[23] Silke Goronzy, Marina Sahakyan, and Wolfgang Wokurek. Is non-native pronunciation modelling necessary ? In EUROSPEECH 2001 [14].

[24] Timothy J. Hazen. A comparison of novel techniques for rapid speaker adaptation. *Speech Communication*, 31:15–33, 2000.

[25] Chao Huang, Tao Chen, and Eric Chang. Adaptive model combination for dynamic speaker selection training. In *Proceedings of the International Conference on Spoken Language Processing*, volume 1, pages 65–68, 2002.

[26] *International Conference on Acoustics, Speech, and Signal Processing*, 2001.

[27] *Proceedings of the International Conference on Spoken Language Processing*, 2000.

[28] R. Kuhn, P. Nguyen, J.-C. Junqua, L. Goldwasser, N. Niedzielski, S. Fincke, K. Field, and M. Contolini. Eigenvoices for speaker adaption. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pages 1771–1774, 1998.

[29] Roland Kuhn, Jean-Claude Junqua, Patrick Nguyen, and Nancy Niedzielski. Rapid speaker adaptation in eigenvoice space. In *IEEE Transactions on Speech and Audio Processing*, volume 8, pages 695–707, 2000.

[30] Fabrice Lauri, Irina Illina, Dominique Fohr, and Filipp Korkmazsky. Using genetic algorithms for rapid speaker adaptation. In EUROSPEECH 2003 [15], pages 1497–1500.

[31] L. Lee and R. Rose. A frequency warping approach to speaker normalization. In *IEEE Transactions on Speech and Audio Processing*, volume 6, pages 49–59, 1998.

[32] C. J. Leggetter and P. C. Woodland. Speaker adaptation of continious density HMMs using multivariate linear regression. In *Proceedings of the International Conference on Spoken Language Processing*, pages 451–454, 1994.

[33] C. J. Leggetter and P. C. Woodland. Flexible speaker adaptation using maximum likelihood linear regression. In *European Conference on Speech Communication and Technology*, pages 1155–1158, 1995.

[34] Karen Livescu and James Glass. Lexical modeling of non-native speech for automatic speech recognition. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 1683–1686, 2000.

[35] S. Matsunaga, A. Ogawa, Y. Yamaguchi, and A. Imamura. Non-native english speech recognition using bilingual english lexicon and acoustic models. In *Proc. of ICASSP*, volume 1, pages 340–343, 2003.

[36] Yoshifumi Onishi and Kenichi Iso. Speaker adaptation by hierarchical eigenvoice. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 576–579, 2003.

[37] M. Padmanabahn, L. R. Bahl, D. Nahamoo, and M. A. Picheny. Speaker clustering and transformation for speaker adaption in large-vocabulary speech recognition systems. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 701–704, 1996.

[38] Ernest J. Pusateri and Timothy J. Hazen. Rapid speaker adaptation using speaker clustering. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 61–64, 2002.

[39] Lawrence R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In *Proceedings of the IEEE*, volume 77, pages 257–286, 1989.

[40] Koichi Shinoda and Chin–Hui Lee. A structural bayes appraoch to speaker adaptation. In *IEEE Transactions on Speech and Audio Processing*, volume 9, pages 276–287, 2001.

[41] Georg Stemmer, Elmar Noeth, and Heinrich Niemann. Acoustic modeling of foreign words in a german speech recognition system. In EUROSPEECH 2001 [14].

[42] Carlos Teixeira, Isabel Trancoso, and Antonio Serralheiro. Accent identification. In *Proc. of ICASSP*, volume 3, pages 1784–1787, 1996.

[43] Frank Thiele and Rolf Bippus. A comparative study of model-based adaptation techniques for a compact speech recognizer. In ASRU 2001 [4], pages 29–32.

[44] Laura Mayfield Tomokiyo. Lexical and acoustic modeling of non-native speech in LVCSR. In *Proc. of the ICSLP*, pages 1619–1622, 2000.

[45] Laura Mayfield Tomokiyo and Alex Waibel. Adaptation methods for non-native speech. In *Proceedings of Multilinguality in Spoken Language Processing*, 2001.

[46] Yu Tsao, Shang–Ming Lee, Fu–Chiang Chou, and Lin–Shan Lee. Segmental eigenvoice for rapid speaker adaptation. In EUROSPEECH 2001 [14], pages 1269–1272.

[47] L. F. Uebel and P. C. Woodland. Improvements in linear transform based speaker adaptation. In ICASSP 2001 [26], pages 49–52.

[48] Ulla Uebler and Manuela Boros. Recognition of non-native german speech with multilingual recognizers. In EUROSPEECH 1999 [13], pages 903–906.

[49] Zhirong Wang, Tanja Schultz, and Alex Waibel. Comparison of acoustic model adaptation techniques on non-native speech. In *Proc. of ICASSP*, pages 540–543, 2003.

[50] Silke Witt and Steve Young. Off-line acoustic modelling of non-native accents. In EUROSPEECH 1999 [13], pages 1367–1370.

[51] G. Zavaliagkos, R. Schwartz, and J. Makhoul. Batch, incremental and instantaneous adaptation techniques for speech recognition. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 676–679, 1995.