Alfonso Martínez del Hoyo Canterla

Design of Detectors for Automatic Speech Recognition

Thesis for the degree of Philosophiae Doctor

Trondheim, May 2012

Norwegian University of Science and Technology
Faculty of Information Technology,
Mathematics and Electrical Engineering
Department of Electronics and Telecommunications

NTNU – Trondheim
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the degree of Philosophiae Doctor
Faculty of Information Technology, Mathematics and Electrical Engineering
Department of Electronics and Telecommunications

ISBN 978-82-471-3336-1 (printed ver.)
ISBN 978-82-471-3337-8 (electronic ver.)
ISSN 1503-8181

Doctoral theses at NTNU, 2012:36

Printed by NTNU-trykk

Abstract

This thesis presents methods and results for optimizing subword detectors in continuous speech. Speech detectors are useful within areas like detection-based ASR, pronunciation training, phonetic analysis, word spotting, etc. Firstly, we propose a structure suitable for subword detection. This structure is based on the standard HMM framework, but in each detector the MFCC feature extractor and the models are trained for the specific detection problem. Our experiments in the TIMIT database validate the effectiveness of this structure for detection of phones and articulatory features.

Secondly, two discriminative training techniques are proposed for detector training. The first one is a modification of Minimum Classification Error training. The second one, Minimum Detection Error training, is the adaptation of Minimum Phone Error to the detection problem. Both methods are used to train HMMs and filterbanks in the detectors, isolated or jointly. MDE has the advantage that any detection performance criterion can be optimized directly. F-score and class accuracy optimization experiments show that MDE training is superior to the MCE-based method.

The optimized filterbanks reflect some acoustical properties of the detection classes. Moreover, some changes are consistent over classes with similar acoustical properties. In addition, MDE-training of filterbanks results in filters significatively different than in the standard filterbank. In fact, some filters extract information from different critical bands.

Finally, we propose a detection-based automatic speech recognition system. Detectors are built with the proposed HMM-based detection structure and trained discriminatively. The linguistic merger is based on an MLP/Viterbi decoder.

Preface

This dissertation is submitted in partial fulfillment of the requirements for the degree of *Philosophiae Doctor* (PhD) at the Norwegian University of Science and Technology (NTNU). My supervisor has been Associate Professor Magne Hallstein Johnsen and my co-supervisor has been Professor Torbjørn Svendsen, both at the Department of Electronics and Telecommunications, NTNU.

The work has been conducted in the period from January 2006 until November 2011. In addition to research activity, the work included the equivalent of one year of full-time course studies, as well as one year of teaching assistant duties. This research project has been funded by a scholarship given by the Faculty of Information Technology, Mathematics and Electrical Engineering, NTNU.

Acknowledgments

First of all I would like to thank my supervisor, Associate Professor Magne Hallstein Johnsen, for his invaluable guidance and support during my research. Moreover, I would like to thank my co-supervisor, Professor Torbjørn Svendsen, for his helpful suggestions.

I would also like to express my gratitude to all the members of the speech group, in particular to Dr. Ingunn Amdal for helpful assistance and collaboration, Trond Skogstad and Timo Mertens for reviewing my articles, and Jarle Bauck Hamar for reviewing my thesis. Further, I am also thankful to my colleagues at the Signal Processing group, specially to Kirsten Marie Ekseth, who has always been helpful and kind to me.

Moreover, I am grateful to my current employer, Atmel Norway, for their flexibility towards my research.

During these years my friends have shared with me many good moments and been supportive when I needed it. I am thankful to Alex, Ellen, Jarle, Jean-Cristophe, Jose, Paolo, Sara, Susana and Tuva.

Finally, I am grateful to my family, specially my father for his support and my mother for being an example of academic excellence. And I would like to thank Caroline for her love, understanding and encouragement.

Trondheim, January 2012
Alfonso Martínez del Hoyo Canterla

Contents

Notation and Symbols

x	Scalars are typeset in non-bold lowercase
$\mathbf{x}$	Vectors are typeset in bold lowercase
$\mathbf{X}$	Matrices are typeset in bold uppercase
$\mathbf{X}^T$	The transpose of $\mathbf{X}$
$\mathbf{X}^{-1}$	The inverse of $\mathbf{X}$
$./$	Element-wise matrix division
$P(\cdot)$	Probability mass
$p(\cdot)$	Probability density

List of Abbreviations

ASR	automatic speech recognition
DBASR	detection-based ASR
GMM	Gaussian mixture model
HMM	hidden Markov model
HTK	hidden Markov model toolkit
MAP	maximum a posteriori
MCE	minimum classification error
MDE	minimum detection error
MPE	minimum phone error
MFCC	Mel-frequency cepstral coefficient
ML	maximum likelihood
MLP	Multi-Layer Perceptron

Chapter 1

Introduction

A *detector* is a binary classifier that discerns between patterns that share a specific quality (the *class*) and the rest (the *anti-class*). Subword detection in continuous speech is then the process of finding segments in the speech signal that belong to a given subword class. For example, a detector for the phone /ih/ would process a speech signal to find two kind of segments: those that belong to /ih/ and those that belong to anything else. Automatic Speech Recognition (ASR) is the process of automatically converting a speech signal into text. Speech detection and recognition are in fact similar problems; the former focuses on separating one speech class from the rest while the latter tries to separate every class from the others.

Detection of phonetic events such as phones and articulatory features has applications within phonetic analysis, word spotting, computer aided pronunciation training (CAPT) and specially in detection-based automatic speech recognition (DBASR). The performance of ASR is still far from the performance of humans in similar tasks. Therefore, there is a search for alternative structures that can reduce this performance gap. In [1] it was argued that the standard approach to ASR is data driven and does not use all available knowledge about speech or language. Therefore, DBASR is an alternative paradigm where speech knowledge sources are integrated in the recognition system in the form of detectors.

The structure of a DBASR system basically consists of a bank of detectors and a linguistic merger. The bank of detectors is built to analyze the speech signal and find information about a set of intermediate classes. These could be detectors for articulatory features, which are speech features based on knowledge about human production and perception of speech. For instance, they can be related to the place and manner of production of the speech signal. Further, the information from the bank of detectors is pro-

cessed by a linguistic merger and an output sequence of linguistic units is hypothesized. Therefore, in DBASR accurate detectors are decisive for the performance of the system.

Phonetics is concerned with describing speech. Specifically, acoustic phonetics studies the acoustic properties of speech sounds, articulatory phonetics deals with the configuration of the vocal tract used to produce speech and linguistic phonetics focuses on how speech sounds are combined in order to make syllables, words and sentences [2]. Subword detectors can be used to automatically transcribe and analyze speech corpora used in phonetic experiments. Further, articulatory feature detectors can produce phonetic transcriptions that are more detailed than broad phonetic transcriptions [3]. Other areas of application are, for instance, forensic phonetics [4], analysis of speech disorders [5] or medical protocols [6].

Keyword spotting is the task of automatically detecting the occurrences of predefined words in a continuous speech signal [7]. This has applications in audio indexing, surveillance, data mining, etc. Early approaches to keyword spotting were based on sliding a window through the speech signal and using dynamic time warping or artificial neural networks to find the keyword. Another approach is the keyword-filler model, where a statistical model is built with three components: a filler model, a background model and a keyword model. The speech signal is then decoded in order to detect keywords. In these approaches keywords are traditionally spotted directly from the speech signal. However, it is also possible to use subword detectors to preprocess the speech signal and spot keywords based on their output information. This can be done, for instance, generating subword-based templates and detecting keywords with dynamic time warping [8], or with statistical models based on subword features [9].

As the world becomes more and more globalized most people need to learn at least a second language, for example English. This is difficult and requires individualized feedback in order to detect and correct errors. Therefore, there is an increasing interest in software tools that can generate this feedback automatically. One of the challenges is to develop applications that can improve the pronunciation of the learner in the new language. In CAPT systems [10] it is useful to measure the overall pronunciation performance on a sentence. However, it is also necessary to detect pronunciation errors and to provide feedback at the phoneme level [11, 12]. Therefore, phone and articulatory feature detectors can also be used to improve the performance of CAPT systems.

In this thesis we have approached subword detection by adapting the standard ASR framework for the two-class problem. A standard ASR sys-

tem extracts acoustic features from the frequency content and the time dynamics of the speech signal. In Mel-frequency cepstral coefficients (MFCCs), the dominant speech representation in ASR, the short-term spectrum is processed with a bank of filters that imitates two important properties of human audition: critical bands and a logarithmic scale for both frequency and loudness. Note that this feature extraction is common to all classes, which means that a single MFCC extraction is performed for every time frame. The frequency content variation over classes is modelled by the class-specific MFCC-densities in a classifier based on the hidden Markov model (HMM), the state-of-the-art statistical model for speech signals. The short time dynamic information of the speech signal is modelled by the MFCC time derivatives and the HMM state structure, and the long time dynamic information is modelled by a language model and lexicon.

In the detector case, the parameters of the feature extractor and the HMMs can be improved for each detector. In the feature extractor a reasonable choice is to keep the MFCC structure, as this has shown to be state-of-the-art preprocessing in ASR for many years. However, the MFCC feature extractor could be optimized for each specific detection problem. For example, one extraction parameter that could be optimized is the MFCC filterbank. The standard filterbank is based on empirical experiments on human auditory perception and it is not clear that it leads to optimal speech recognition or subword detection. Moreover, a filterbank optimized for a specific class would probably reflect some of the typical frequency content of that class. The HMM state-density parameters in the detector are also good candidates for detector-specific optimization. A possible approach is to use discriminative training techniques, as they are commonly applied to ASR systems.

In the next section of this introductory chapter we give an overview of research related to our work. After that, Section 1.2 lists the major contributions of this thesis. Finally, Section 1.3 presents an outline of the rest of the thesis.

1.1 Related Work

Design of subword detectors and their evaluation have been studied in connection with detection-based ASR systems [13, 14, 15]. In the ASAT (Automatic Speech Attribute Transcription) [1] and SIRKUS (Spoken Information Retrieval by Knowledge Utilization in Statistical speech processing) projects, there have been a number of successful speech recognition systems that were based on detectors, for example a lattice re-scoring ap-

proach [16], a phone recognizer [17] or a large-vocabulary continuous speech recognizer [18]. In the latter system a bank of MLP-based detectors generated articulatory feature posteriors for each speech frame. Then an MLP-based evidence merger mapped the scores into phone posteriors, which were combined into word lattices by a weighted finite state machine. The last module in this system rescored those lattices using a language model. Further, it is worth mentioning that HMM were used as "attribute detectors" in [19, 20]. However, as far as we understood, what they built were in fact HMM-based classifiers for each exclusive group of attributes, which is conceptually different than our approach. In addition, their method would not allow a class specific detector optimization.

Landmark-based ASR [21, 22, 23, 24] is a related knowledge-based approach to speech recognition. Landmarks identify parts of an utterance where articulatory features are most salient. In a first step landmarks can be detected with landmark-specific acoustic features. After that, in order to identify articulatory features, further analysis can be done in the utterance regions specified by the detected landmarks. In a final step, this information is integrated by some probabilistic framework in order to find linguistic units such as phones or words.

There is a number of discriminative training methods that have been successfully applied to ASR, for example Maximum Mutual Information (MMI) [25], Minimum Phone Error (MPE) [26] and Minimum Classification Error (MCE) [27]. The relationship between these methods was studied in [28]. Both segment-based and string-based MCE were used in [13] to train detectors for the task of isolated and continuous detection of subwords. Even if in some cases MCE training led to performance improvements, they argued for a modified version for detector training. The reason was that the standard MCE training focused on increasing the accuracy of all classes and then there was no guarantee that the performance for the target class would improve. A possible solution would be to apply weighted MCE [29], which is a training criterion suitable for tasks with non-uniform error costs.

Previous work in data-driven filterbank optimization for speech recognition includes the following studies. Firstly, joint training of filterbank and the back-end classifier, for example a prototype-based distance classifier [30] or a HMM-based isolated word recognition system [31]. Secondly, in [32, 33] the filterbank was optimized to produce robust features. Further, there have been work on filterbank design using MPE [34], Linear Discriminant Analysis [35] or minimum entropic distance [36]. In [37] they studied filterbank parameters such as shape and center frequencies, and then they optimized them using the simplex method.

Filterbank optimization is one approach to acoustic feature optimization. In the following we present some other approaches. Firstly, linear and non-linear transformations can be applied to MFCC features, for example [38, 39]. An study on MCE training of linear feature transformation was presented recently in [40]. Secondly, discriminative training of the feature extractor and the classifier was studied in [41]. Thirdly, MLPs have been applied to the front-end of speech recognizers. The basic approach has been the Tandem system, where an MLP estimates phone posteriors from MFCCs and a HMM-based decoder uses a decorrelated version of these posteriors as input features. A review of MLP-based approaches was presented recently in [42]. Further, work on MPE-trained feature transformations was presented in [43, 44]. In addition, a combination of MPE-based transformations and MLP features was proposed in [45]. Finally, optimization of classification performance for a subset of classes using a discriminative feature transformation was studied in [46].

1.2 Contributions of This Thesis

This thesis provides a study on detection of subwords in continuous speech. Our main contributions are as follows. Firstly, we propose a subword detector structure based on the standard ASR framework. Secondly, this thesis presents two discriminative training methods that can be applied to our detector structure. Thirdly, we provide a study on detector-specific MFCC filterbank optimization. Parts of this work have been published in [47, 48].

1.2.1 A Subword Detection Structure Based on Standard ASR

Our approach to subword detection design has been to adapt the standard continuous speech recognition framework, which basically consists of a MFCC extractor and a HMM-based decoder. Further, the parameters of this detector structure can be optimized in each detector to improve the performance for the detection class. In this thesis we have focused on phone and articulatory-feature detectors. However, the detector structure is suitable for other subwords, for example syllables.

A common approach to subword detector design is to use frame-based classifiers, for example using Multi-Layer Perceptrons (MLPs). Detectors based on HMMs have previously been proposed, for example [13]. However, this mainly consisted in the use of two HMMs with similar complexity, one for the class and one for the anti-class. In addition, HMM-based detectors

have mostly been used for detection of isolated speech segments in ASR re-scoring applications. By contrast, the anti-class models in our detector structure are built with all the HMMs of the subwords that belong to the anti-class. Our motivation for this was that we assumed that a more complex anti-class model would lead to better segmentation in the continuous speech case. In addition, this type of anti-class model is flexible and does not require additional training.

1.2.2 Discriminative Training Methods for Subword Detection

Subword detectors have previously been trained applying the same discriminative training techniques that are used for automatic speech recognition. Therefore, in this thesis we propose two novel discriminative training methods that focus specifically on subword detectors. The first approach to subword detector training is a modification of the standard string-based MCE framework, which was motivated by the results presented in [13]. Basically, our modification consists in selecting the frames that are included in the computation of the gradient so that the performance of the detection class is improved.

The second method for subword detector training is based on the MPE framework and, therefore, we call it Minimum Detection Error (MDE) training. As far as we know, MPE has not previously been applied to subword detector optimization. This novel technique is capable of directly optimizing the F-score or any other performance measure for detectors, which means that MDE can be applied to a wide range of applications.

1.2.3 Detector-Specific MFCC Filterbank Optimization and Analysis

The two discriminative training methods that we have developed for detector optimization have been applied to optimize the MFCC filterbank in our detector structure. In fact, filterbank training is a novel technique for optimization of subword detectors. Further, the MFCC filterbanks have been trained either isolated or jointly with the HMMs used in the class and anti-class models in the detector. In addition, this thesis presents an analysis of the optimized filterbanks and we find that some of the changes in the filter shapes reflect known acoustical properties of the detection classes.

1.3 Outline

The first part of the thesis focuses on subword detector design, optimization and applications. Chapter 2 is a short introduction to HMMs and standard ASR systems. In Chapter 3 we present a detector structure for subword units. Chapter 4 focuses on discriminative training and describes two methods for detector training. In Chapter 5 we introduce detection-based ASR and present a system based on our detector structure.

The second part of this thesis focuses on the experiments with subword detectors. The experimental settings are described in Chapter 6. Experiments with subword detector optimization are presented in Chapters 7 and 8. Chapter 9 describes the experiments where detection-based ASR systems are built with optimized detectors. Finally, Chapter 10 presents the conclusions and suggestions for future work.

Part I

Subword Detectors: Design and Optimization

Chapter 2

Standard Automatic Speech Recognition

Speech is the most natural and important communication channel for humans. Automatic Speech Recognition (ASR) deals with the problem of automatically identifying the string of words embedded in an acoustic speech signal. In the previous chapter we mentioned that our approach to subword detection consists on adapting the standard ASR framework for the two class problem. Therefore, in this chapter we give a short introduction to standard ASR systems. Firstly, ASR is described as a pattern-matching problem. After that, we give an overview of the main components in a standard recognizer. The last two sections focus on the feature extraction module and the acoustical models.

2.1 ASR as a Pattern Recognition Problem

The current approach to ASR is the one of statistical pattern recognition [49, 50]. Given a spoken utterance S we find the string of words $\hat{W}$ that maximizes the *a posteriori* probability $P(W|S)$. That is

$$\hat{W} = \arg\max_{W} P(W|S) \; . \tag{2.1}$$

This is called the *maximum a posteriori* (MAP) decision rule and is optimal in the sense that it minimizes the error rate. However, the optimality is only valid if the true statistical distributions are known. In practice this is not true and, therefore, some parametrical distributions are usually assumed. The parameters are then estimated using training data and the MAP decision rule is evaluated with these estimated distributions. This is known as

plug-in MAP decision rule [51].

In practice, acoustical features $\mathbf{X}$ are extracted from the speech signal S. Applying Bayes' rule, the corresponding posterior probability can be rewritten as

$$P(W|\mathbf{X}) = \frac{p(\mathbf{X}|W)P(W)}{p(\mathbf{X})} \,. \tag{2.2}$$

The denominator is independent of the class W and, therefore, it can be discarded in the MAP context. The numerator consists of two terms. The first one gives the acoustic likelihood for the class and is usually modeled by hidden Markov models (HMMs). The second term gives the *a priori* probability of the class W and is usually approximated by a language model. Speech is usually split up into hierarchically organized linguistic units such as sentences, words, syllables, phonemes, etc. Typically HMMs are used to model a chosen basic unit, for example phonemes, and higher order unit models, for instance sentences, can be composed from the models of the chosen basic units using a the language model and a lexicon.

ASR tasks can be divided into two main categories. The first one is isolated speech classification, where the segment borders between speech units are known. The second group is continuous speech recognition, which is a more difficult problem because both the segmentation and the corresponding string of units are unknown. Each basic unit is modeled as a HMM and the likelihood of each string is computed as the likelihood of the concatenation of the corresponding basic units. This thesis focuses on detection of speech units in continuous speech and, therefore, the rest of the chapter considers continuous speech recognition.

2.2 Overview of a Standard HMM-Based ASR System

A block diagram of a standard system is shown in Figure 2.1. The input to the system is a speech signal $s(t)$ and the output is a hypothesized string of linguistic units. Following the notation of the previous section, the first step is to extract a set of features $\mathbf{X}$ from the speech utterance S. The system then finds the string of linguistic units with maximum $P(W|\mathbf{X})$ using a decoder, an acoustic model, a language model and a lexicon. The following subsections describe the modules shown in Figure 2.1.

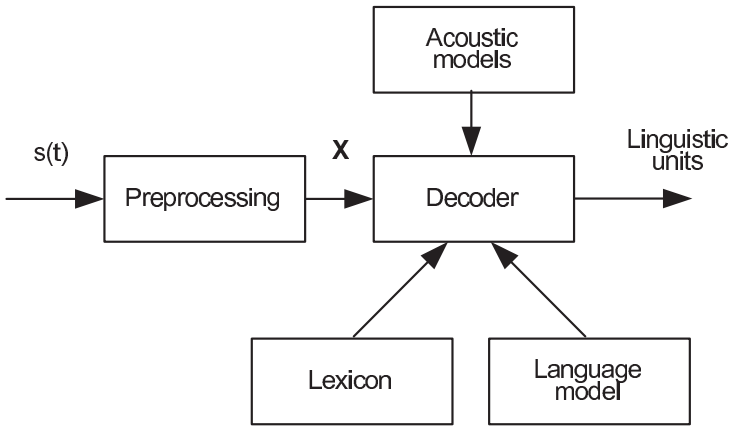


Figure 2.1: Block diagram of a standard HMM-based ASR system

2.2.1 Feature Extraction

The input to this block is the speech signal $s(t)$ and the output a sequence of feature vectors $\mathbf{X}$. An ideal feature extractor would discard all the information that refers to the speaker and the environment. The remaining information should provide good discrimination between linguistic units.

In the standard approach to feature processing, the speech signal is divided in short speech frames. Each frame can be modeled as the realization of a corresponding stationary process. Usually some information about the power spectrum is extracted for each frame. The most popular choice of features, Mel-frequency cepstral coefficients (MFCCs), is described in detail in Section 2.4. Other features that are commonly used, especially for noisy speech, are perceptual linear predictive (PLP) features [52].

2.2.2 Acoustic Model and Basic Units

The acoustic model provides the likelihood of the acoustic features $\mathbf{X}$ given the linguistic unit that is modelled, i.e. the basic unit. In standard ASR the acoustic model is a HMM, which is reviewed in more detail in Section 2.3. There are a number of choices for the basic units, for example phonemes, syllables and words:

- From [53, p. 37], "the term *phoneme* is used to denote any of the minimal units of speech sound in a language that can serve to distinguish one word from another. We conventionally use the term *phone* to denote a phoneme's acoustic realization"

- A *syllable* is an intermediate unit between phones and the word level, usually centered around vowels.

- A *word* is a higher order unit that can be written or spoken and has an agreed-upon meaning.

The task usually determines which basic unit to model. In some small-vocabulary applications HMM word modelling usually leads to the best system performance. However, when the vocabulary increases in size it is not practical to use words as basic units and smaller units, for instance phonemes, are modelled instead. A more detailed discussion is given in Section 5.2.1 in the context of detection-based ASR systems.

2.2.3 Language Model

A language model gives the probability of linguistic units that are modelled. The simplest language models are the *0-gram* and the *unigram*, where the probability of a unit is independent of the context (the surrounding linguistic units). In the former each unit has the same probability and, for this reason, it is also referred to as *uniform* model. However, in more complex models the probability of a given linguistic unit is dependent on the context. For example, the *bigram* and *trigram* models assume that the probability of a unit depends, respectively, on the previous one or two linguistic units.

Usually the language model is completed with the use of a *grammar* and a *lexicon*. The grammar determines the possible transitions between linguistic units. A simple example is a unit loop where each unit can be followed of any other. The lexicon (also referred to as *dictionary*) provides one or several ways to construct higher order units from basic units.

The specific implementation depends on the task, the basic linguistic unit that is modelled, etc. For a small-vocabulary word recognizer that uses phone HMMs, the implementation of the language model could start with an unconstrained loop over all words as a grammar, where the word transition probabilities could be given by a word bigram model. Further, this could be combined with a lexicon with the phone transcription of each word. Another example could be the implementation of a phone recognizer that uses phone HMMs. In this case no lexicon would be required, the grammar could be an unconstrained loop over all phones governed by phone trigram probabilities.

2.2.4 Decoder

The decoder uses the acoustical and language models to find a hypothesis for the string of linguistic units in the output of the system. This sequence of linguistic units is the one that matches best the input features according to the plug-in MAP decision rule. The language model and the HMMs for the basic units are used to build a network of HMM states. A given path (state sequence) in this network corresponds to a string of linguistic units. The size of the state network is usually large and the best path has to be found using an efficient search algorithm. The Viterbi algorithm (for instance [53, p. 388]) is based on dynamic programming and it is commonly applied for this task.

In some applications it can be useful to find not only the best path, but also several several competing hypotheses that can be further processed. This could be in the form of a *N-best* list [54], which corresponds to the N paths in the network with highest likelihood. In addition, each hypothesis is given together with the most likely segment borders. An N-best list can be useful, for example, in some large vocabulary ASR applications where a fast system provides N best hypothesis and a slower but more accurate system re-scores them to generate a final output string of linguistic units. Another way of representing a set of competing hypothesis are *lattices*. This is basically a graph where nodes correspond to times and edges correspond to speech segments. Therefore, in a lattice a complete path corresponds to a sentence hypothesis. Lattices can be generated from N-best lists and *vice versa*.

2.2.5 Performance Evaluation

The performance of the system is found by comparing the correct transcription of the input speech signal with the output of the system, i.e. the hypothesized string of linguistic units. The strings are aligned according to the minimum Levenshtein distance and the number of hits H, substitutions S, insertions I and deletions D are found. The *accuracy* of a system A_t is defined as

$$A_t = \frac{H - I}{H + D + S} = \frac{H - I}{N} \,, \tag{2.3}$$

where $N = H+D+S$ is the number of segments in the correct transcription. We have added the subindex t to emphasize that this accuracy is the *total* accuracy of the system, meaning that it considers the accuracy for all classes. This is further discussed in Section 3.3.

In the case of isolated speech recognition there are no deletions or insertions and the performance is then called *correctness*:

$$C_t = \frac{H}{H+S} \,. \tag{2.4}$$

This performance measure is commonly used in continuous speech recognition to complement the information given by the accuracy.

Finally, it could be added that in some cases there is a linguistic unit mapping previous to the alignment. This is done because sometimes modelling a more detailed linguistic unit set can lead to better results compared to modelling the linguistic unit set of interest.

2.3 The Hidden Markov Model

The hidden Markov model (HMM) is a simple statistical model that is commonly used to model the speech signal. From [27, p.259], when humans speak the articulatory apparatus modulates air pressure and flow to produce the sounds that constitute the speech signal. Even if speech is a time-varying signal, there are short time regions where the signal can be considered to be a stationary process. This quasi-stationarity is a consequence of physical constraints in the articulatory apparatus, which makes that the articulatory configuration cannot change dramatically more than ten times per second. In addition, there is usually a certain dependency between sounds in the speech signal that occur after each other, which means that speech is not a memoryless process. The HMM accommodates these properties and, therefore, it is used in the standard ASR framework.

Given the set of observations $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2 \dots \mathbf{x}_T\}$, each vector $\mathbf{x}_t$ is assumed to have been drawn according to a probability density function. There are two important assumptions in HMMs:

- Output-independence assumption: the probability density function is assumed to depend only on the actual state of the model s_t.
- Markov assumption: even though the state variable is unknown, the sequence of states during time is assumed to follow a random process called first-order Markov chain. This means that the state at a given time t, referred to as s_t, depends only on the state at the previous time $t-1$, referred to as s_{t-1}.

The components of a HMM are the state probability density function, the transition probabilities between states and the initial state distribution:

$$b_i(\mathbf{x}_t) = p(\mathbf{x}_t|s_t = i) \ , \tag{2.5}$$

$$a_{ij} = P(s_t = j|s_{t-1} = i) \ , \text{ and} \tag{2.6}$$

$$\pi_i = P(s_1 = i) \ . \tag{2.7}$$

Figure 2.2 shows a HMM with an oriented structure. This is called a left-to-right HMM and it is commonly used in ASR. For instance, for phoneme modelling it is standard to use left-to-right HMMs with three states.

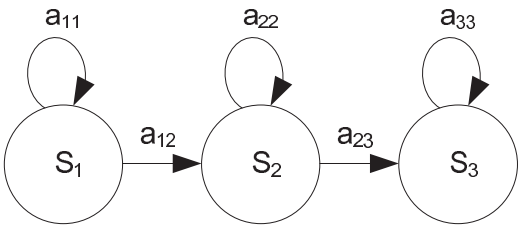


Figure 2.2: Left-to-right HMM

A commonly used state output probability function is the Gaussian mixture model (GMM). For a state s we have that

$$b_s(\mathbf{x}) = \sum_{m=1}^{M} c_{sm} b_{sm}(\mathbf{x}) \tag{2.8}$$

$$= \sum_{m=1}^{M} \frac{c_{sm}}{(2\pi)^{D/2}|\mathbf{\Sigma}_{sm}|^{1/2}} \, e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_{sm})^T \mathbf{\Sigma}_{sm}{}^{-1}(\mathbf{x}-\boldsymbol{\mu}_{sm})} \ . \tag{2.9}$$

where M is the number of mixtures in the GMM, $b_{sm}(\mathbf{x})$ is a Gaussian probability density function with parameters $\boldsymbol{\mu}_{sm}$ (the mean vector) and $\mathbf{\Sigma}_{sm}$ (the covariance matrix), D is the dimension of the observation vector $\mathbf{x}$ and c_{sm} are the mixture coefficients. These mixtures coefficients have to follow the following constraints:

$$c_{sm} > 0 \ , \quad \forall m \tag{2.10}$$

$$\sum_{m} c_{sm} = 1 \tag{2.11}$$

In the rest of this section we use $\mathbf{\Lambda}$ to denote all parameters in a HMM: transition probabilities, GMM means and covariances, mixture coefficients, etc.

2.3.1 Evaluation

The evaluation problem consists on finding the likelihood of the set of observations $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2 \dots \mathbf{x}_T\}$ given by a HMM with parameters $\mathbf{\Lambda}$. Considering the state sequence $\mathbf{S} = \{s_1, s_2, ...s_T\}$, we have that

$$p(\mathbf{X}|\mathbf{\Lambda}) = \sum_{\text{all } \mathbf{S}} p(\mathbf{S}, \mathbf{X}|\mathbf{\Lambda}) = \sum_{\text{all } \mathbf{S}} P(\mathbf{S}|\mathbf{\Lambda}) p(\mathbf{X}|\mathbf{S}, \mathbf{\Lambda}) \ , \tag{2.12}$$

where the sum is over all possible values of the state sequence $\mathbf{S}$. Given the Markov assumption, the probability of a given state sequence is the product of the corresponding state transition probabilities:

$$P(\mathbf{S}|\mathbf{\Lambda}) = \pi_{s_1} \prod_{t=2}^{T} a_{s_{t-1} s_t} \ . \tag{2.13}$$

In addition, because of the output-independence assumption we have that

$$p(\mathbf{X}|\mathbf{S}, \mathbf{\Lambda}) = \prod_{t=1}^{T} b_{s_t}(\mathbf{x}_t) \ . \tag{2.14}$$

Applying these two results to Eq. 2.12, the likelihood of the observation $\mathbf{X}$ results in

$$p(\mathbf{X}|\mathbf{\Lambda}) = \sum_{\text{all } \mathbf{S}} \pi_{s_1} b_{s_1}(\mathbf{x}_1) \prod_{t=2}^{T} a_{s_{t-1} s_t} b_{s_t}(\mathbf{x}_t) \ . \tag{2.15}$$

However, the direct computation of this probability is in practice impossible because the total number of state sequences grows exponentially with T. Fortunately, it is possible to compute it efficiently with a recursive algorithm known as the forward algorithm, given for example in [53, p. 387].

2.3.2 Decoding

In the decoding problem we are interested in finding the optimal state sequence, i.e. the state sequence $\mathbf{S}^*$ with the highest likelihood, given a set of observations and a HMM. We saw in Section 2.2.4 that this is essential in an ASR system: the decoder uses the language and acoustic models to build a large network of HMM states and the best path determines the output of the system. Given that the network is large, an efficient search algorithm such as the Viterbi algorithm finds the optimal state sequence $\mathbf{S}^*$ and its corresponding likelihood $\mathbf{S}^*$, $p(\mathbf{S}^*, \mathbf{X}|\mathbf{\Lambda})$. In addition, the likelihood of $\mathbf{S}^*$ is commonly used as an approximation to the likelihood in Eq. 2.15:

$$p(\mathbf{X}|\mathbf{\Lambda}) \approx p(\mathbf{S}^*, \mathbf{X}|\mathbf{\Lambda}) = \max_{\mathbf{S}} \pi_{s_1} b_{s_1}(\mathbf{x}_1) \prod_{t=2}^{T} a_{s_{t-1}s_t} b_{s_t}(\mathbf{x}_t) \ . \tag{2.16}$$

Because the number of possible state sequences can be very large, it may seem that approximating the sum in 2.15 by the highest term is not a good approximation. However, in practice most terms in the sum are much smaller than the highest one and the approximation is in fact useful.

2.3.3 Training

The traditional training approach is the following one: given a HMM and training data (a set of labelled observations) the parameters that best describe the data are estimated, which is also referred to as Maximum Likelihood (ML) learning. Specifically, this corresponds to finding $\mathbf{\Lambda}$ such that $p(\mathbf{X}|\mathbf{\Lambda})$ is maximized for a training sequence $\mathbf{X}$. Mathematically this can be written as

$$\mathbf{\Lambda}_{\mathrm{ML}} = \arg\max_{\mathbf{\Lambda}} p(\mathbf{X}|\mathbf{\Lambda}) \tag{2.17}$$

The problem here is that the state sequence is unknown and, therefore, ML training cannot be applied directly. The standard solution is to apply a version of the expectation-maximization (EM) algorithm adapted to HMM. This training method is known as the Baum-Welch algorithm or as the forward-backward algorithm (see for instance [53, p. 389]). Section 4.1 brings more details on training algorithms and parameter optimization methods.

2.4 Mel-Frequency Cepstral Coefficients

As in all pattern recognition problems, the choice of acoustic features is critical in speech recognition. MFCCs [55] are the dominant speech representation because of their good performance, especially in clean environments. Figure 2.3 shows the block diagram for a MFCC feature extractor. The input is the speech signal and the output is a vector of cepstral coefficients $\mathbf{x}$.

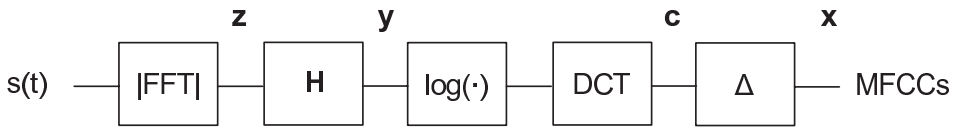


Figure 2.3: Block diagram of the MFCC feature extractor.

According to [56], there are three main parts in the MFCC extractor:

- Short-term Fourier analysis of the speech signal
- Auditory-motivated modifications of the spectrum
- Projection on cosine basis

The motivation for the short-time analysis is the following one. Even if speech is a non-stationary process, it can be considered stationary over sufficiently short-time intervals. Therefore, the Fourier transform of a short segment should give a good spectral representation of the segment. In the first module, the speech signal is pre-emphasized and thereafter divided into frames which are multiplied by a window. Typical choices are 10 ms frame shift and 25 ms Hamming windows. Then a Fast Fourier Transform (FFT) with N_{FFT} points is computed for each windowed frame and the magnitude or the squared magnitude is applied. The resulting vector is referred as $\mathbf{z}$ in Figure 2.3.

The resulting spectrum is modified to model some properties of human hearing. Firstly, there is a non-uniform frequency resolution in human perception. Secondly, humans seem to integrate the energy of signals that fall into some bands, which are referred to as *critical bands*. Moreover, the bandwidth of these critical bands increases with the center frequency. These properties are simulated in feature extraction by processing the short-time spectrum with a filterbank. The next block in Figure 2.3 is then a bank of N_{CH} critical filters (or channels) that performs the linear mapping $\mathbf{y} = \mathbf{H}\,\mathbf{z}$.

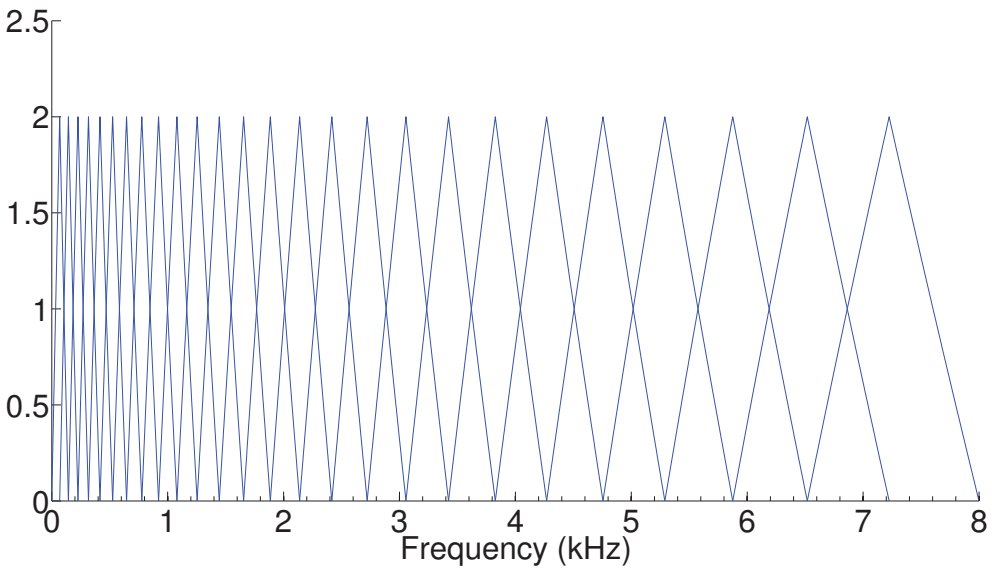


Figure 2.4: Standard filterbank in the MFCC feature extractor.

Each component y_i represents the averaged spectral energy in the corresponding critical band of the filter. The center frequencies of the filters are spaced according to a frequency scale motivated by how humans perceive sounds. Usually this is the Mel frequency scale, which is approximated by

$$M(f) = 1125 \ln(1 + \frac{f}{700}) \,, \tag{2.18}$$

where f is the linear frequency (in Hz). Moreover, the bandwidths of the critical-band filters increase with the frequency according to the Mel scale. Further, the shape of the filters is such that the averaging prioritizes the center of the critical band over the borders and, therefore, triangular or Gaussian filters are typically used. Figure 2.4 shows a triangular shaped Mel-scale filterbank.

The final perceptually motivated transformation of the spectrum is done in the next block in Figure 2.3. A logarithmic transformation is done to the energy averages output by the filterbank. This operation models human loudness sensitivity.

The next block is a discrete cosine transform (DCT), which maps the

log-energies to the cepstral domain. As discussed in [56], projection onto a cosine basis approximately decorrelates vectors generated by a first-order Markov process. This means that the covariance matrices in the GMMs state distributions of the HMM can be considered to be diagonal matrices. The elements of a unitary DCT matrix are given by

$$D_{ij} = a_{ij} \cos\left[\frac{\pi(2j-1)}{2N_{\text{CH}}}(i-1)\right], \tag{2.19}$$

where $i, j = 1, 2, \ldots N_{\text{CH}}$, and

$$a_{ij} = \begin{cases} \sqrt{\frac{1}{N_{\text{CH}}}} & i, j = 1 \\ \sqrt{\frac{2}{N_{\text{CH}}}} & i = j \\ 1 & \text{otherwise.} \end{cases} \tag{2.20}$$

The DCT block performs the mapping $\mathbf{c} = \mathbf{D} \log \mathbf{y}$, which transforms a vector of N_{CH} components into $\mathbf{c}$, a vector of N_{CH} cepstral coefficients. However, due to the energy compaction property of the DCT, most discriminative information lies in the first N_{CEP} coefficients. It is customary to truncate the cepstral vector, i.e. those coefficients are kept and the rest are discarded, which is known as cepstral smoothing of the log-magnitude spectrum. In fact, most of the variation lies on the 0^{th} cepstral coefficient, which is a measure of the speech frame energy. This means that most of the variation in the speech spectrum is due to variation in the average energy. For this reason, some other energy measurement, for example power, can be used either in addition or as a replacement.

Finally, it is known that appending the first and second order time derivatives (delta and acceleration coefficients) improves the performance of the system considerably [57]. The final block in Figure 2.3 computes the first and second order time derivatives and appends them to $\mathbf{c}$.

For instance, for a speech signal sampled at 16 kHz it is common to use $N_{\text{CH}} = 26$ filters and use the $N_{\text{CEP}} = 13$ first cepstral coefficients. Further, after the addition of the time derivatives we end up with a feature vector $\mathbf{x}$ with 39 components.

2.5 Summary

Our method for subword detection is based on the standard speech recognition framework and, therefore, we have reviewed standard ASR systems in

this chapter. The input of a speech recognizer is the speech signal and the output is a hypothesis of linguistic units. The speech signal is processed by a feature extractor to generate an observation: a sequence of acoustic feature vectors with discriminative information. Further, a decoder finds the sequence of linguistic units that best matches the observation using acoustic models, language models, a grammar and a lexicon. This process usually involves searching for the best path in a state network and, therefore, an efficient search algorithm is required. The performance of a speech recognizer is commonly evaluated aligning the output with a reference transcription and computing the accuracy, a measure that combines hits, substitutions, deletions and insertions.

In the standard speech recognition framework, MFCCs are the state-of-the-art feature representation of speech and HMMs are commonly used to model the speech signal. This thesis focuses on the MFCC feature extraction and the HMM and, therefore, they were reviewed in more detail.

We discussed the components of a HMM: state probability density function (commonly implemented with GMMs), transition probabilities (we saw that the left-to-right configuration is commonly used in ASR) and initial distributions. After that we discussed the implied statistical assumptions: output-independence and Markov assumptions. In addition, we gave an overview of the evaluation of a HMM with the forward algorithm, Viterbi decoding and training with Baum-Welch.

The MFCC feature extractor computes cepstral coefficients from the speech signal as follows. Firstly, the speech signal is divided in frames and a window function is applied. After that a FFT is computed and a bank of filters is applied to model the critical band sensitivity of human hearing. Further, human loudness sensitivity is modelled by a logarithm. Finally, a DCT transform is applied to decorrelate the coefficients, and time derivatives are appended to the resulting cepstral coefficients.

Chapter 3

Design of Subword Detectors

In Chapter 1 we saw that a detector is a binary classifier that discerns between patterns that share a specific quality (the class) and the rest (the anti-class). In this thesis, we consider detectors that find the occurrences of a type of subword in the speech signal. For example, a detector for the phone /ih/ will process the speech signal to find segments that belong to that phone. In this case, /ih/ segments are referred to as *class segments*, and segments that belong to the other phones are referred to as *anti-class segments*. One of the challenges found in detection is that most patterns belong to the anti-class. This has to be considered in the training algorithm.

In this chapter we address the design of subword detectors and their evaluation. Training and optimization methods will be presented in Chapter 4. The outline of the chapter is as follows. Firstly, Section 3.1 describes two approaches to detection of subword units: frame-based and segment-based detectors. Secondly, in Section 3.2 we propose a detection structure based on standard ASR. Finally, Section 3.3 discusses the evaluation of subword detectors and presents the criteria that have been considered in this thesis. Parts of this work were also presented in [47, 48].

3.1 Detection Strategies

This section presents a number of strategies for the design of subword detectors. Firstly, we discuss the advantage that detectors have in feature extraction compared with a standard ASR system. After that, frame-based and segment-based detectors are described. This categorization follows the work in [13].

Figure 3.1: Block diagram of a frame-based detector.

3.1.1 Acoustic Feature Extraction

The feature extractor module is the first part in a detector. In a standard ASR system the feature extraction is common to all classes, but in subword detection there can be a specific feature extractor in each detector. This is in principle an advantage because it is possible to process the speech signal and extract features that are optimal for the specific class vs. anti-class problem in each subword detector. However, it is of course possible to use the same features in all the detectors.

One possibility is to use a standard feature extraction method and optimize some of its parameters. For example, the filterbank, window length or time shift could be optimized in a MFCC-based feature extractor (see Section 2.4). Another option is to use features based on phonetic knowledge of the specific detection class, for instance formants, voiced onset time, zero-crossing rate, etc. In any case the extraction parameters (and thus the features) can be optimized independently or jointly with the detector structure.

3.1.2 Detector Based on Frames

A frame-based detector classifies speech frames individually. Figure 3.1 shows a block diagram of this type of detector. The input of the detector is the speech signal and the output is a sequence of labelled frames. The first module is a feature extractor that finds discriminative information for the speech frames.

The next module is a binary classifier that processes the extracted features to generate scores. In practice there are two common strategies for the score generation. The first one consists on generating a single class score for each frame. The second one is to generate scores both for the anti-class and the class.

There are many possible structures for the binary classifier, for example

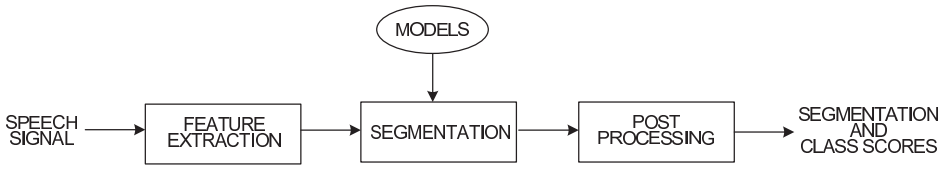


Figure 3.2: Block diagram of a segment-based detector.

Multi-Layer Perceptron (MLP) or, more generally, artificial neural networks (ANN), support vector machine or Gaussian mixture model (GMM). The output of frame-based detectors are usually posterior probabilities. In the case of using GMM-based class densities, there is usually a class GMM and an anti-class GMM. Each of them provides a likelihood, which can be further processed if a posterior probability is required.

The last step in the detector is to use the scores from the binary classifier in a decision rule. A simple method is to apply a threshold to the class score. However, a decision rule could consider both class and anti-class scores, for instance a threshold for the score ratio.

It should be noted that in applications such as detection-based ASR, the decision rule step is commonly skipped. In those systems, the generated detector scores are further processed by some type of linguistic merger in order to find a string of linguistic units. More details about this are given in Chapter 5.

3.1.3 Detectors Based on Segments

Segment-based detectors have a speech signal as an input and they output a sequence of labelled segments. Figure 3.2 shows the block diagram of this type of detector. As in frame-based detectors, the first module is the feature extractor. However, in the case of segment-based detection, all the speech feature frames are processed in order to find class segments. This means that, in addition to the segment scores given by the detector, the segmentation is important and has to be considered as well. In segment-based detectors the segmentation can be found with a decoding algorithm and statistical models for the class and the anti-class. Segment-based detectors were found to perform better than frame-based detectors in [13]. In their experiments they found that ANN detectors had a problem with

output scores fluctuation. In the rest of the section, we focus on detectors where the class and the anti-class models are built with HMMs. However, there are other options, for example Dynamic Bayesian Networks, see for instance [58].

HMM-based detectors output the segmentation of the speech signal and the likelihood of each segment. The class model usually consists of a single HMM for the detection class. We consider two possible strategies to build the anti-class model in a HMM-based detector, similar to those methods described in [16]. Firstly, a simple method is to use a single HMM for the anti-class. This HMM could have the same or higher complexity than the class model and it could be specific for each detector or common to all detectors ("background" model). The second strategy is to build the anti-class model with a combination of the HMMs of all the other classes, or with only those belonging to the main competing classes. In this case, these HMMs are combined with a grammar, for instance parallel transitions corresponding to each HMM. As an example, let us suppose that there are three subwords and we are interested in building a HMM-based detector for the first subword. The class model can be built with a single HMM that is trained with the data of the first subword. In the first method, the anti-class model is a single HMM, which is trained with the data of all the other classes (the second and third subwords). In the second method, the anti-class model is built with two HMMs that are trained, respectively, with the data of the second and third subwords.

For the segmentation the decoder requires a network (grammar) that define the transitions between the class and the anti-class models. This could include the use of transition penalties to weight some paths of the network. A simple network is a loop between the class and the anti-class models, which is shown in Figure 3.3. Other strategies are, however, possible; for instance in [14] only the class model was used to segment the speech signal and, after that, class and anti-class models were Viterbi-aligned to each segment in order to compute a log-likelihood ratio.

After the segmentation it is possible to apply a postprocessing algorithm. This includes pruning of segments in order to eliminate insertions, which could be done using knowledge-based methods, for example temporal information of attribute models [59], or other methods such as, for instance, verification [60, 61].

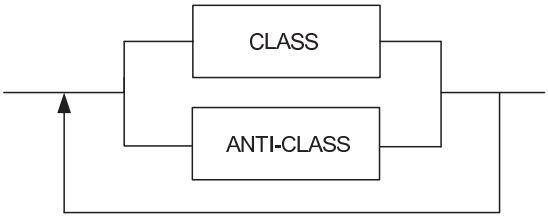


Figure 3.3: An example of grammar network for segment-based detectors.

3.2 Proposed Structure for Subword Detection

The block diagram of the proposed structure for subword detection is shown in Figure 3.4. There are two ways to justify this detection structure. Firstly, it is a special case of the segment-based detectors strategy described in Section 3.1.3. Secondly, the chosen method for the design of subword detectors is an adaptation of the standard ASR framework for a two-class problem. In the following we explain the chosen detector structure as a special case of a segment-based detector. In the end of the section we compare the detector structure with a standard ASR system.

The first module in a segment-based detector is the feature extractor, which was discussed in Section 3.1.1. As shown in Figure 3.4, the chosen feature extraction method was a MFCC module (see Figure 2.3) optimized for the specific detector. There are a number of extraction parameters that can be optimized, but we focused on the filterbank for a number of reasons. Firstly, the standard filterbank is based on empirical experiments on human auditory perception and it is not clear that it leads to optimal detection regardless of the specific phonetic event to detect, structure of the back-end classifier, speaker, environment, etc. In fact, several studies have shown that it can be optimized for ASR by data-driven techniques, for example [30]. Secondly, the shape of the standard filterbank can probably be modified to extract information that is relevant to discriminate class and anti-class. Thus, the resulting filterbank would reflect some of the typical frequency content of the detection class. Thirdly, the blocks in the MFCC feature extractor have a cascade connection (see Figure 2.3), which means that the filterbank is well suited for gradient based discriminative training. This could be done by training the filter parameters, for instance bandwidths and spacing, or by training all the elements in the filterbank matrix $\mathbf{H}$. Note that MFCC parameters such as the window length or time shift would be

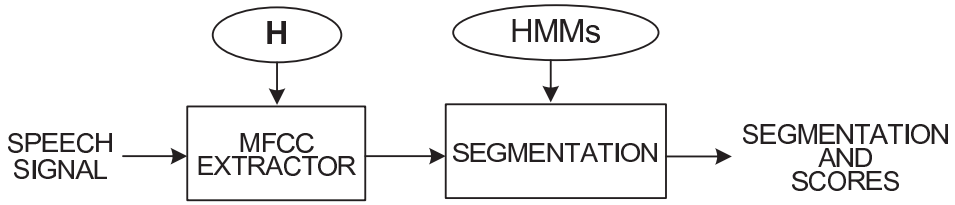


Figure 3.4: Block diagram of the proposed detector structure.

more more difficult to train discriminatively. The feature extractor in our detector structure is then a MFCC extractor where the filterbank is specific to the detector. This is the meaning of the filterbank block pointing to the MFCC block in Figure 3.4.

The next module in the proposed detector structure is the segmentation. In this work we have focused on detectors for phonemes and articulatory features (AFs). The class model in phone detectors was built with a single HMM. AF detectors are commonly trained with a database transcribed at the phone level and a mapping to AFs (Appendix A describes this in more detail). We chose to model the class in AF detectors with a combination of the HMMs of the phones for which the AF is active (this is similar to the approach in [20]). In both cases, the anti-class models were built with a combination of all the other phone HMMs, i.e. those that belong to the anti-class. These class and anti-class models are used by a decoder (see Section 2.2.4) to generate an output segmentation. The grammar used in the decoder was a loop between the class and anti-class models. In addition, the class and anti-class models are implemented with their corresponding phone HMMs in parallel. This means that the proposed network in the decoder results in a phone loop both for the case of phone and AF detectors. Therefore, the segmentation module in the proposed detection structure is a decoder that uses a detector-specific set of phone HMMs. This is the meaning of the HMMs block pointing to the segmentation block in Figure 3.4. Figure 3.5 shows an example where there are C phones and we consider the detector for an AF that is active for phone 1 and phone 2. The class model is then built with HMM_1 and HMM_2 in parallel, and the anti-class model is built with all the other models, i.e. $\{\text{HMM}_i\}_{i=3}^{C}$, in parallel. It can be seen that this decoding network is equivalent to all models $\{\text{HMM}_i\}_{i=1}^{C}$ in parallel, i.e. a phone loop.

As with the MFCC feature extractor, these class and anti-class models should also be optimized in each detector. The state-density parameters in the phone HMMs are good candidates for detector-specific discriminative training. Therefore, the proposed optimization method for our detection structure is to train the phone HMMs and the MFCC filterbank matrix discriminatively. All detectors can be initialized with phone HMMs trained with Maximum Likelihood and a standard filterbank in the MFCC extraction module. The discriminative optimization of filterbanks and HMMs is addressed in Chapter 4.

In the beginning of this section we mentioned that our detection structure can be regarded as an adaptation of the standard ASR framework for the class vs. anti-class problem. Comparing the proposed detector structure with the standard system described in Section 2.2, we find the following common points. Firstly, the feature extractor in the detection structure is indeed a MFCC module where the standard filterbank is replaced by a filterbank optimized for the specific detector. Secondly, given how the class and anti-class models were implemented, the decoder is basically the same as in a standard phone recognizer that uses a phone loop as the grammar. The difference is that the HMMs in the proposed detector structure are optimized so that the performance for one sub-group of the phones is improved.

Finally, note that even if this subword detector structure has been applied to detection of phones and AFs, the structure is general and could be applied to detect other subwords such as syllables. Moreover, this structure could in principle be adapted for word spotting in small vocabulary tasks were each word can be modelled with a HMM.

3.3 Evaluation of Subword Detectors

This final section addresses the evaluation of the subword detector structure that has been proposed. In Section 2.2.5 we discussed performance measures for ASR systems. There we saw that in continuous speech recognition the segment sequence is aligned with a reference transcription to find hits (H), substitutions (S), deletions (D) and insertions (I). In the case of detection of subwords there are four outcomes: *hits* (correct classifications of class segments), *correct rejections* (correct classifications of anti-class segments), *false alarms* (incorrect class segments) and *misses* (class segments that were not detected). The detected transcription can be aligned with a reference and a confusion matrix generated:

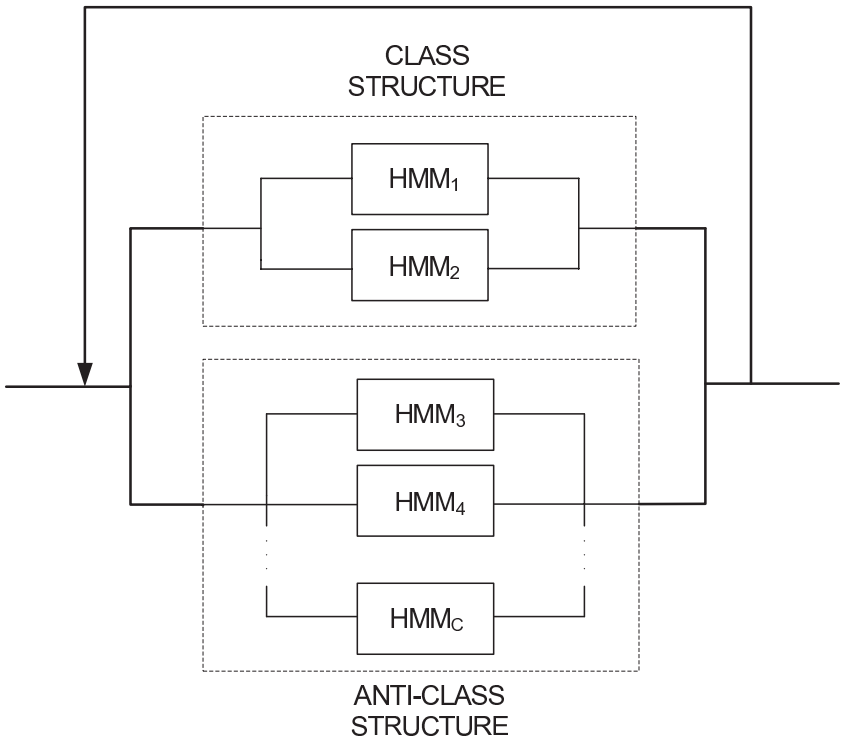


Figure 3.5: A decoding network for the proposed detector structure.

$$\left[\begin{array}{cc|c} H_c & S_c & D_c \\ S_{ac} & H_{ac} & D_{ac} \\ \hline I_c & I_{ac} & \end{array}\right],$$

where the subindexes "c" and "ac" refer to respectively the class and the anti-class. The relevant outcomes are then hits (H_c), misses (S_c and D_c) and false alarms (S_{ac} and I_c).

Note that in the proposed detector structure the decoder outputs a segmentation with subword labels (for example /ph1/', /ph2/, etc.) and not detector labels (for instance "class" and "anti-class"). The evaluation is as follows. Firstly, the output segmentation at the phone level is aligned with the reference transcription. Secondly, phone labels are mapped to detector labels and H_c, S_c, etc. are computed. Thirdly, any of the detector

evaluation criteria discussed below can be applied.

An ideal detector would identify correctly all class segments (which implies no misses) without raising any false alarms. Further, the segment borders would be highly accurate. However, in practice we must accept a trade off between hits, false-alarms and misses, as well as having imperfect time information.

Detection performance criteria should consider that the number of class segments in a detection application is usually much lower than the number of anti-class segments. If we wanted to evaluate a detector using the same criterion as in ASR systems (see Equation 2.3), the accuracy of the detector would result in

$$A_t = \frac{H_c + H_{ac} - I_c - I_{ac}}{N_c + N_{ac}} \,. \tag{3.1}$$

However, this criterion is not suited for detection applications because in practice a high A_t can be achieved even if the number of class hits, H_c, is low. This can be explained by the fact that usually $H_c \ll H_{ac}$, $I_c \ll I_{ac}$ and $N_c \ll N_{ac}$, where $N_c = H_c + S_c + D_c$ (the number of class segments in the reference transcription) and similarly for N_{ac}.

The following criteria are commonly used to score detectors: *precision*, *recall* and *F-score* (or F-measure). These are performance measures that can be used in pattern recognition applications where one of the classes is more important than the rest [62]. Note that there are other possible criteria, for example the DET curve [63]. Precision (P) is the relationship between hits and the number of class segments in the detected transcription:

$$P = \frac{\text{hits}}{\text{hits} + \text{false alarms}} = \frac{H_c}{M_c} \,, \tag{3.2}$$

where $M_c = H_c + S_{ac} + I_c$. Recall (R) is the relationship between hits and the number of class segments in the reference:

$$R = \frac{\text{hits}}{\text{hits} + \text{misses}} = \frac{H_c}{N_c} \,. \tag{3.3}$$

F-score (F) is the harmonic mean of P and R, which results in

$$F = \frac{2PR}{P + R} = \frac{2H_c}{N_c + M_c} \,. \tag{3.4}$$

In addition, in this thesis we propose to use the accuracy of the detector as evaluation criterion. This *class accuracy* A_c is defined as

$$A_c = \frac{H_c - I_c}{N_c} = R - \frac{I_c}{N_c} \,, \tag{3.5}$$

which is the commonly used ASR accuracy limited to include only class segments. Note that the class accuracy is more restrictive than R because it considers the inserted class segments I_c.

The choice of evaluation criterion has to consider the target application of the detectors as well. For example, in the context of DBASR it is important not to miss candidates and recall is usually prioritized over precision [15]. In this thesis we have considered the F-score and the class accuracy. The former leads to a balanced performance between hits, misses and false alarms. The latter leads to detectors that should perform well in DBASR. However, note that these methods do not consider segmentation accuracy directly. Some work on performance evaluation combining detection and segmentation performance was done in [64].

3.4 Summary

In this chapter we discussed the design of subword detectors. There are two main groups of detectors: frame-based and segment-based detectors. Detectors based on frames classify each speech frame individually and can be implemented with binary classifiers such as for example an MLP. Detectors based on segments process all speech frames with a decoding algorithm, class and anti-class models in order to produce a sequence of class and anti-class segments, border information and scores. In addition, we proposed a segment-based detector that can be regarded as a standard ASR system where the filterbank and the HMMs are discriminatively trained to improve the performance of the detector. Finally, we discussed the evaluation of subword detectors. We argued that ASR system evaluation criteria such as the commonly used accuracy are not valid for detectors. Therefore, we presented four measures that are suitable for detection evaluation: precision, recall, F-score and class accuracy.

Chapter 4

Discriminative Training of Subword Detectors

In this chapter we present two discriminative training methods for the detector structure presented in the previous chapter. The first section discusses the motivation for discriminative optimization methods and, in addition, it introduces some background on optimization and our discriminative framework. In the second section we present the standard embedded Minimum Classification Error training and how we have modified it in order for it to be applied to subword detector training. The third section describes Minimum Detection Error (MDE) training, the adaptation of Minimum Phone Error training for subword detection. Parts of this work have been published in [47, 48].

4.1 Discriminative Training Methods

4.1.1 Motivation

Recall from Section 2.1 that posterior probabilities are required for an optimal classification decision. The optimal decision rule in the classifier is then given by

$$\hat{W} = \arg\max_{W} p(W|X) \,, \tag{4.1}$$

where W is any possible word string and X is the observation vector. As discussed in [27], this transforms the classifier design problem into a distribution estimation problem. However, there are a number of issues with this approach. Firstly, in practice the distributions are unknown and we

have to find a suitable model for them, which is usually limited by mathematical tractability. In the case of speech recognition, HMM is a simple method to model the speech signal (see Section 2.3) but it is not the true distribution and, therefore, the Bayes error cannot be achieved. Secondly, the parameters of the distribution model have to be estimated from training data. In order to obtain good parameter estimates an estimation method is required and, in addition, training data of sufficient size has to be collected and labelled. This is specially difficult and expensive in speech recognition.

In supervised training, the conventional approach to classifier design is known as the *generative approach*. This consists in trying to model $P(W, X)$, the joint generative model of observations and classes, which is usually done by modeling the likelihood of the observation $P(X|W)$ and the language model $P(W)$. The classifier implements then the following decision rule:

$$\hat{W} = \arg\max_{W} p(X|W)p(W) \ . \tag{4.2}$$

The traditional method of parameter estimation, Maximum Likelihood (ML), was discussed in Section 2.3.3. ML estimation does not necessarily lead to an optimal performance in the classifier because of two reasons. Firstly, this method leads to optimal parameters for the estimated distributions with respect to data description. Given the mismatch between the HMM and the actual (unknown) speech distribution, there is no connection between data description optimality and classification optimality. Secondly, even if we had very good models for the real distributions, recall that training data is often limited and inadequate.

The *discriminative approach* to classifier design aims at modelling directly the posterior distributions $P(W|X)$, which is used in the decision rule given by Eq. 4.1. This is the case in Multi-Layer Perceptron, support vector machine and logistic regression [65]. This approach has the advantage that the training algorithm focuses on those regions that are important for the classification task. By contrast, the generative approach aims at describing the data accurately and this can lead to focus on regions that may be unimportant for the classification task. However, the generative approach has also strengths, for example it leads to better classifier performance when training data are limited. A further discussion can be found in [66, 67, 68].

There are a number of approaches to combine the benefits of generative and discriminative classifiers. One of them is *discriminative training*, where the parameters of a generative classifier are trained using a discriminative criterion, that is a function related to the performance of the classifier. In discriminative training there are two critical aspects: design of the perfor-

mance function and the choice of the optimization method that is used to optimize this performance function. Some of the most popular discriminative training techniques for speech recognition are Maximum Mutual Information (MMI), Minimum Classification Error (MCE) and Minimum Phone Error (MPE). These methods have different motivations but they are closely related, as shown in [69, 70]. However, there are many other discriminative training techniques, for example Direct Error Rate Minimization [71].

In the following we introduce the mathematical framework and the optimization method that we have considered. After that, we present two discriminative training algorithms that can be applied to train subword detectors.

4.1.2 Framework

The training data is a set of sentences $\mathcal{X} = \{\mathbf{X}_1, \ldots, \mathbf{X}_K\}$ and their labels $\mathcal{L} = \{L_{10}, \ldots, L_{K0}\}$, where the index 0 indicates that this is the correct transcription. Given the set of system parameters $\mathbf{\Lambda}$, a set of N competing segmentations $\{L_{kj}\}_{j=1}^{N}$ can be generated for each sentence $\mathbf{X}_k$ (see Section 2.2.4). A sentence has a number of frames: $\mathbf{X}_k = \{\mathbf{x}_1, \ldots, \mathbf{x}_{T(k)}\}$ and each frame is labelled at the model i and state s level:

$$L_{kj} = \{(i_{j1}, s_{j1}),\ \ldots\ , (i_{jT(k)}, s_{jT(k)})\}\ . \tag{4.3}$$

Note that the notation has been relaxed so that the dependence on k is implicit.

Given a transcription L_{kj} and the parameters set $\mathbf{\Lambda}$, the log-likelihood of a sentence $\mathbf{X}_k$ is

$$g_j(\mathbf{X}_k; \mathbf{\Lambda}) = \log p(\mathbf{X}_k | L_{kj}, \mathbf{\Lambda}) = \sum_{t=1}^{T(k)} \log a_{s_{j,t-1} s_{jt}} + \sum_{t=1}^{T(k)} \log b_{i_{jt} s_{jt}}(\mathbf{x}_t)\ , \tag{4.4}$$

where t is the frame index, $a_{s_{j,t-1} s_{jt}}$ is the transition probability between states $s_{j,t-1}$ and s_{jt}, and $b_{i_{jt} s_{jt}}(\mathbf{x}_t)$ is the state probability density function given by a Gaussian mixture model (see Eq. 2.9) for model i_{jt} and state s_{jt}. In the following, the notation is relaxed so that the dependence on j is implicit for i_{jt} and s_{jt}.

We consider a scalar function that is related to the performance of the system and thus it is referred to as *performance function*:

$$J(\mathbf{\Lambda}) = \sum_k f(\mathbf{X}_k; \mathbf{\Lambda})\ , \tag{4.5}$$

where $f()$ is a scalar function that measures the performance of the system for one sentence. This function will be determined by the specific discriminative method that is considered. However, in our work we have considered discriminative methods that lead to performance functions of the following form:

$$J(\mathbf{\Lambda}) = \sum_k f(g_0(\mathbf{X}_k; \mathbf{\Lambda}), \ldots, g_N(\mathbf{X}_k; \mathbf{\Lambda})) \; . \tag{4.6}$$

4.1.3 Optimization Methods

An introduction to optimization methods in the context of discriminative training can be found in [70, Section 1.5]. Growth Transformation (GT) is a family of methods for parameter optimization. These methods are iterative and estimate the new set of parameters $\mathbf{\Lambda}^{n+1}$ using a transformation of the actual set of parameters $\mathbf{\Lambda}^n$:

$$\mathbf{\Lambda}^{n+1} = T(\mathbf{\Lambda}^n) \; . \tag{4.7}$$

This transformation has the property that the performance function grows in its value unless $\mathbf{\Lambda}^{n+1} = \mathbf{\Lambda}^n$. Examples of these re-estimation methods are Expectation Maximization (EM) or the Extended Baum-Welch (EBW) algorithm, which is focused towards the optimization of HMM parameters. The advantages of EBW are monotone convergence and close-form parameter updating formulae. However, GT-based methods require a performance function with rational form and this leads to a more complex implementation.

GT-based methods are related to gradient optimization methods. In its simplest form, a gradient-based method updates the parameters iteratively following the following rule:

$$\mathbf{\Lambda}^{n+1} = \mathbf{\Lambda}^n + \alpha \frac{\partial J(\mathbf{\Lambda}^n)}{\partial \mathbf{\Lambda}} \; , \tag{4.8}$$

where α is referred to as *learning rate*. The Generalized Probabilistic Descent (GPD) is a popular method from this family. More recently, other methods like Quickprop or Resilient propagation have been proposed. Even if these methods do not have the desirable monotone convergence property of EBW, they can be easily implemented and in practice their performance in discriminative training makes them a valid alternative to EBW.

4.1.4 Gradient-Based Optimization of HMMs and Filterbanks

In this thesis we have chosen gradient-based optimization because of its simpler implementation. These methods require the computation of the gradient of the performance function in each iteration. Considering Eq. 4.6 and the chain rule of multivariate functions we have that

$$\frac{\partial J(\mathbf{\Lambda})}{\partial \mathbf{\Lambda}} = \sum_{k,j} \frac{\partial f(g_0(\mathbf{X}_k;\mathbf{\Lambda}) \dots g_N(\mathbf{X}_k;\mathbf{\Lambda}))}{\partial g_j(\mathbf{X}_k;\mathbf{\Lambda})} \frac{\partial g_j(\mathbf{X}_k;\mathbf{\Lambda})}{\partial \mathbf{\Lambda}} \ . \tag{4.9}$$

The first term will be dependent on the specific discriminative training that is implemented, which determines the function $f()$. However, the second term shows that the computation of the gradient of the log-likelihood with respect to the parameters to optimize is common to all the methods that we have considered. Therefore, in the rest of this section we will compute this term for the parameters of interest.

In this work we have focused on the optimization of the HMMs and the filterbank matrix in the MFCC feature extractor. In principle it is possible to optimize the means, covariances, transition probabilities and mixture weights in the HMMs. However, optimization of the means usually brings most of the performance improvement. The gradient of the log-likelihood with respect to $\boldsymbol{\mu}_{ism}$ can be obtained applying the chain rule of differential calculus:

$$\begin{aligned} \frac{\partial g_j(\mathbf{X}_k;\mathbf{\Lambda})}{\partial \boldsymbol{\mu}_{ism}} &= \sum_t \frac{\partial \log b_{i_t s_t}(\mathbf{x}_t)}{\partial \boldsymbol{\mu}_{ism}} \\ &= \sum_t \delta_{kjtis} \sum_m \frac{c_{ism} \cdot b_{ism}(\mathbf{x}_t)}{b_{is}(\mathbf{x}_t)} \ \boldsymbol{\Sigma}_{ism}{}^{-1}(\mathbf{x}_t - \boldsymbol{\mu}_{ism}) \ , \end{aligned} \tag{4.10}$$

where m is the index for the mixtures in model i and state s and δ_{kjtis} indicates that L_{kj} classifies frame $\mathbf{x}_t$ as model i and state s. Mathematically this can be written as

$$\delta_{kjtis} = \begin{cases} 1 & \text{if } (i_t, s_t) = (i, s) \\ 0 & \text{otherwise.} \end{cases} \tag{4.11}$$

Note that only a subset of the training data is used for the training of $\boldsymbol{\mu}_{ism}$.

There are a number of methods that have been proposed to train the filterbank in the MFCC feature extractor. One possible option is to constrain the optimization by assuming a parametric shape for the filters. However,

in this thesis we chose to represent the filterbank by a matrix $\mathbf{H}$ and train each matrix element. The gradient of the log-likelihood with respect to $\mathbf{H}$ can be obtained applying the chain rule of differential calculus:

$$\begin{aligned}\frac{\partial g_j(\mathbf{X}_k;\mathbf{\Lambda})}{\partial \mathbf{H}} &= \sum_t \frac{\partial \log b_{i_t s_t}(\mathbf{x}_t)}{\partial \mathbf{H}} \\ &= \sum_{t,m} \frac{c_{i_t s_t m} \cdot b_{i_t s_t m}(\mathbf{x}_t)}{b_{i_t s_t}(\mathbf{x}_t)} \frac{\partial \mathbf{x}_t}{\partial \mathbf{H}} {\mathbf{\Sigma}_{i_t s_t m}}^{-1}(\boldsymbol{\mu}_{i_t s_t m} - \mathbf{x}_t) \ , \end{aligned} \quad (4.12)$$

where m is the index for the mixtures in model i_t and state s_t. Note that in this case all the frames in the training data are used to train the coefficients of the filterbank matrix.

The gradient of the feature vector with respect to the filterbank matrix can be computed as follows. Firstly, we could divide $\mathbf{x}_t$, $\mathbf{\Sigma}^{-1}$ and $\boldsymbol{\mu}$ into static cepstrum, first derivative and acceleration parts:

$$\mathbf{x}_t = \begin{bmatrix} \mathbf{c}_t \\ \mathbf{d}_t \\ \mathbf{a}_t \end{bmatrix} \qquad \boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_c \\ \boldsymbol{\mu}_d \\ \boldsymbol{\mu}_a \end{bmatrix} \qquad \mathbf{\Sigma}^{-1} = \begin{bmatrix} \mathbf{\Sigma}_c^{-1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Sigma}_d^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{\Sigma}_a^{-1} \end{bmatrix} \ , \quad (4.13)$$

where $\mathbf{c}_t$ is the cepstral vector, $\mathbf{d}_t$ is the vector of first time derivatives and $\mathbf{a}_t$ is the vector of second order time derivatives, etc. In addition, we have considered that the covariance matrix $\mathbf{\Sigma}^{-1}$ is diagonal and, therefore, $\mathbf{\Sigma}_c^{-1}$, $\mathbf{\Sigma}_d^{-1}$ and $\mathbf{\Sigma}_a^{-1}$ are diagonal matrices as well. In addition, we have denoted by $\mathbf{0}$ a matrix with all its entries being zero, i.e. a null matrix. Further, in Section 2.4 we saw that the vector of cepstral features is computed as

$$\mathbf{c}_t = \mathbf{D} \log \mathbf{y}_t \ , \quad (4.14)$$

where

$$\mathbf{y}_t = \mathbf{H}\mathbf{z}_t \ . \quad (4.15)$$

The method to compute the gradient of $\mathbf{x}_t$ with respect to $\mathbf{H}$ is general. However, for simplicity let us consider that the first order time derivatives are computed as

$$\mathbf{d}_t = \mathbf{c}_{t+2} - \mathbf{c}_{t-2} \ , \quad (4.16)$$

and the second order time derivatives as

$$\mathbf{a}_t = \mathbf{d}_{t+2} - \mathbf{d}_{t-2} \ . \tag{4.17}$$

Section B.1 shows that the gradient of the observation vector $\mathbf{x}_t$ with respect to the filterbank matrix $\mathbf{H}$ is given by the following expression:

$$\begin{aligned}\frac{\partial \mathbf{x}_t}{\partial \mathbf{H}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu} - \mathbf{x}_t) = (\mathbf{w}_t^c./\mathbf{y}_t)\mathbf{z}_t^T + (\mathbf{w}_t^d./\mathbf{y}_{t+2})\mathbf{z}_{t+2}^T - (\mathbf{w}_t^d./\mathbf{y}_{t-2})\mathbf{z}_{t-2}^T + \\ (\mathbf{w}_t^a./\mathbf{y}_{t+4})\mathbf{z}_{t+4}^T - (\mathbf{w}_t^a./\mathbf{y}_{t-4})\mathbf{z}_{t-4}^T - 2(\mathbf{w}_t^a./\mathbf{y}_t)\mathbf{z}_t^T\end{aligned} \tag{4.18}$$

where the vectors $\mathbf{w}_t^c$, $\mathbf{w}_t^d$ and $\mathbf{w}_t^a$ are defined as

$$\begin{aligned}\mathbf{w}_t^c &= \mathbf{D}^T\boldsymbol{\Sigma}_c^{-1}(\boldsymbol{\mu}_c - \mathbf{c}_t) \\ \mathbf{w}_t^d &= \mathbf{D}^T\boldsymbol{\Sigma}_d^{-1}(\boldsymbol{\mu}_d - \mathbf{d}_t) \\ \mathbf{w}_t^a &= \mathbf{D}^T\boldsymbol{\Sigma}_a^{-1}(\boldsymbol{\mu}_a - \mathbf{a}_t) \ .\end{aligned} \tag{4.19}$$

Note that the optimization of the filterbank matrix should be constrained to positive values. When constrains are in the form of inequalities a common method for optimization is to transform the related variables to fulfill the constrain. In our case we have that $h_{ij} > 0$ and then we could transform $\tilde{h_{ij}} = \exp(h_{ij})$. Given that the exponential function is monotonically increasing, the optimization of one variable corresponds to the solution of the other. As mentioned in [70], it is important to notice, however, that there could be sensitivity problems in the solution when using this technique.

In this thesis we have chosen Resilient propagation (RPROP) [72] to optimize the performance function. However, note that the proposed framework is independent of the specific gradient-based optimization algorithm. For completeness, the following describes this algorithm. RPROP is an efficient learning scheme that adapts the weight step directly based on local gradient information. Each parameter that is trained with RPROP has an individual update value Δ. If the derivative of the performance function changes sign, a local optimum was jumped over and the update-value is decreased by α^-. If the derivative of the performance function maintains the sign, the update-value is increased by α^+. The parameter to train is modified according to the sign of the derivative, with the exception that if a local optimum has been jumped over the previous parameter value is

reverted. Algorithm 4.1 describes one training iteration for any parameter $\mathbf{\Lambda}$. The sign change in the sign function with respect to [72] is because in this thesis we are interested in maximizing the performance function.

Algorithm 4.1 Resilient propagation (RPROP) algorithm

if $\frac{\partial J}{\partial \Lambda}(n-1) \cdot \frac{\partial J}{\partial \Lambda}(n) > 0$ **then**
  $\Delta(n) = \min(\Delta(n-1) \cdot \alpha^+, \Delta_{\max})$
  $\Delta\Lambda(n) = \text{sign}(\frac{\partial J}{\partial \Lambda}(n)) \cdot \Delta(n)$
  $\Lambda(n+1) = \Lambda(n) + \Delta\Lambda(n)$
else if $\frac{\partial J}{\partial \Lambda}(n-1) \cdot \frac{\partial J}{\partial \Lambda}(n) < 0$ **then**
  $\Delta(n) = \max(\Delta(n-1) \cdot \alpha^-, \Delta_{\min})$
  $\Lambda(n+1) = \Lambda(n) - \Delta\Lambda(n-1)$
  $\frac{\partial J}{\partial \Lambda}(n) = 0$
else if $\frac{\partial J}{\partial \Lambda}(n-1) \cdot \frac{\partial J}{\partial \Lambda}(n) = 0$ **then**
  $\Delta\Lambda(n) = \text{sign}(\frac{\partial J}{\partial \Lambda}(n)) \cdot \Delta(n)$
  $\Lambda(n+1) = \Lambda(n) + \Delta\Lambda(n)$
end if

In the rest of this chapter we present two discriminative training techniques that can be applied to subword detectors. Recall from the beginning of this section that the first term in Eq. 4.9 is specific to each training algorithm and this is computed in the next sections for each method. The second term is, however, common to all methods and is given by Eqs. 4.10 for the HMM means, and by 4.12 and 4.18 for the filterbank matrix.

4.2 Minimum Classification Error Training

The motivation of MCE [27, 73, 74] is to build a performance function based on the how a classifier operates. Specifically, the decision rule of the classifier is approximated with a smooth function of the system parameters $\mathbf{\Lambda}$ and the training data. The performance of the classifier is then estimated as a soft count over the correct decisions in the training set. Optimization of the performance function leads to an increase in the number of correct decisions in the classifier. MCE can be applied at the level of different speech units, for example phone, word or sentences. In our work we build detectors that are based on phone HMMs and, therefore, it may seem logical to build a performance function that counts the number of correctly

recognized phones. However, we have focused on detection of subwords in continuous speech and it has been reported that in this context string-level MCE (also referred to as MCE at the sentence level or embedded MCE) is a more effective way of improving phone accuracies [73]. A simple explanation is that improving the number of correct sentences improves indirectly the number of correctly recognized phones and, in addition, in this method the phone models are modified taking in consideration the sorrounding phones, which is critical in continuous speech recognition. Therefore, we have chosen string-based MCE as the basis for the subword detection training method. It is important to notice that even if string-based MCE is more effective than phone-based MCE, there is a mismatch between the performance function (sentence accuracy) and the evaluation performance (phone accuracy). In the rest of the section we present the MCE framework and discuss the modifications of the standard method in order for it to be applied to subword detector training.

One of the keys of MCE training is to express in a smooth function the decision rule implemented by the classifier. If we ignore the language model we have that the decision rule in the classifier is

$$L(\mathbf{X}_k) = L_{kj} \Longleftrightarrow g_j(\mathbf{X}_k;\mathbf{\Lambda}) = \max_u g_u(\mathbf{X}_k;\mathbf{\Lambda}) \ . \tag{4.20}$$

The operation of the classifier can be approximated with a smooth function refered to as classification measure $d(\mathbf{X}_k;\mathbf{\Lambda})$. In this thesis we have considered the same function as in [27], which has the following form

$$d(\mathbf{X}_k;\mathbf{\Lambda}) = g_0(\mathbf{X}_k;\mathbf{\Lambda}) - \frac{1}{\eta}\log\left\{\frac{1}{N}\sum_{j=1}^{N}\exp\left[g_j(\mathbf{X}_k;\mathbf{\Lambda})\cdot\eta\right]\right\}, \tag{4.21}$$

where η is a positive number and the discrepancy in sign with respect to [27] is because they considered a misclassification measure. There are, however, other possible functions that can be used as classification measure [70]. When $\eta \to \infty$ the second term approaches $\max_{j=1:N} g_j(\mathbf{X}_k;\mathbf{\Lambda})$ and then $d(\mathbf{X}_k;\mathbf{\Lambda}) > 0$ means a correct decision. For smaller values all the segmentations are taken into consideration resulting in a non-zero decision threshold.

The classification measure $d(\mathbf{X}_k;\mathbf{\Lambda})$ can be embedded into a smoothed zero-one function. Far from the threshold of the zero-one function correct decisions will have a near unity value and errors will have a near zero value. In this work with have considered the sigmoid function:

$$\mathrm{sigm}\left[d(\mathbf{X}_k;\mathbf{\Lambda})\right] = \frac{1}{1+\exp\left[-\gamma d(\mathbf{X}_k;\mathbf{\Lambda})+\theta\right]} \ . \tag{4.22}$$

where the values of θ and γ determine the threshold and, therefore, these parameters should be optimized during training. Further, the performance function is defined as

$$J(\mathbf{\Lambda}) = \sum_k \text{sigm}\left[d\left(\mathbf{X}_k; \mathbf{\Lambda}\right)\right] \ , \tag{4.23}$$

which is a soft count of the correctly classified sentences in the training set.

We need to compute the gradient of J with respect to the system parameters $\mathbf{\Lambda}$. Applying the chain rule of differential calculus we obtain the following expression:

$$\frac{\partial J(\mathbf{\Lambda})}{\partial \mathbf{\Lambda}} = \sum_k \text{sigm}'[d(\mathbf{X}_k; \mathbf{\Lambda})] \frac{\partial d(\mathbf{X}_k; \mathbf{\Lambda})}{\partial \mathbf{\Lambda}} \ . \tag{4.24}$$

Further, the gradient of the classification measure with respect to the parameters is given by

$$\frac{\partial d(\mathbf{X}_k; \mathbf{\Lambda})}{\partial \mathbf{\Lambda}} = \sum_{j=0}^{N} \beta_j(\mathbf{X}_k; \mathbf{\Lambda}) \frac{\partial g_j(\mathbf{X}_k; \mathbf{\Lambda})}{\partial \mathbf{\Lambda}} \ , \tag{4.25}$$

where $\beta_j(\mathbf{X}_k; \mathbf{\Lambda})$ is defined as

$$\beta_j(\mathbf{X}_k; \mathbf{\Lambda}) = \begin{cases} 1 & \text{if } j = 0 \\ \frac{-\exp[g_j(\mathbf{X}_k; \mathbf{\Lambda}) \cdot \eta]}{\sum_{i=1}^{N} \exp[g_i(\mathbf{X}_k; \mathbf{\Lambda}) \cdot \eta]} & \text{otherwise.} \end{cases} \tag{4.26}$$

Finally, recall from Section 4.1.4 that the gradient of the log-likelihood with respect to the HMM model means and the filterbank matrix is given, respectively, by Eqs. 4.10, 4.12 and 4.18.

4.2.1 MCE Training for Detectors

The theory derived in the previous section is used to train the model means and the filterbank matrix to increase the number of correctly classified sentences (see Eq. 4.23). Recall that this improvement in sentence accuracy improves indirectly the overall phone accuracy but not necessarily the detector performance [13]. Moreover, if we applied the standard MCE training to the proposed subword detector structure, there would be no difference between detectors. That is of course considering that the same initial values in the optimization for the models and the filterbank in all detectors.

Therefore, the standard string-based MCE training method was modified so that it can be applied to training of subword detectors. In our modified MCE training for detectors the frames that are used to train the means and filterbank matrix are a subset of the frames that would be used in the standard MCE method. In addition, this subset depends on the detector class c. In the standard method we train $\boldsymbol{\mu}_{ism}$ with all frames that belong to class i and state s in at least one of the $N+1$ sentences. However, in our modified MCE training for the detector for target class c it is required that in addition those frames belong to class c in at least one of the $N+1$ hypotheses. Note that this makes no difference when training the model for the target class $\boldsymbol{\mu}_{csm}$. In other words, the means in the anti-class models, $\boldsymbol{\mu}_{ism},\ i \neq c$, are trained with a fraction of the data, while the means of the class model $\boldsymbol{\mu}_{csm}$ are trained as in the standard MCE training. Applying this modification to Eq. 4.10 leads to the following expression:

$$\frac{\partial g_j(\mathbf{X}_k;\boldsymbol{\Lambda})}{\partial \boldsymbol{\mu}_{ism}} = \sum_t \delta_{kjtis} \cdot \delta_{ktc} \sum_m \frac{c_{ism} \cdot b_{ism}(\mathbf{x}_t)}{b_{is}(\mathbf{x}_t)}\ {\boldsymbol{\Sigma}_{ism}}^{-1}(\mathbf{x}_t - \boldsymbol{\mu}_{ism})\ , \tag{4.27}$$

where δ_{kjtis} is given by Eq. 4.11 and δ_{ktc} indicates whether there exists one L_{kj} that classifies frame $\mathbf{x}_t$ as the target class c. Mathematically this can be written as

$$\delta_{ktc} = \begin{cases} 1 & \text{if } \exists\, j : i_t = c \\ 0 & \text{otherwise.} \end{cases} \tag{4.28}$$

Similarly, applying the modification to Eq. 4.12 leads to

$$\frac{\partial g_j(\mathbf{X}_k;\boldsymbol{\Lambda})}{\partial \mathbf{H}} = \sum_t \delta_{ktc} \sum_m \frac{c_{i_t s_t m} \cdot b_{i_t s_t m}(\mathbf{x}_t)}{b_{i_t s_t}(\mathbf{x}_t)}\ \frac{\partial \mathbf{x}_t}{\partial \mathbf{H}}\ {\boldsymbol{\Sigma}_{i_t s_t m}}^{-1}(\boldsymbol{\mu}_{i_t s_t m} - \mathbf{x}_t)\ . \tag{4.29}$$

Figure 4.1 shows and example where two frames, $[\mathbf{x}_1, \mathbf{x}_2]$, are classified as $L_0 = [c_1, c_2]$ and $L_1 = [c_2, c_3]$ by transcriptions 0 and 1, respectively. To make it simpler, states are not considered. Consider the training of $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$ in the detector for c_1. In the standard embedded MCE training, $\boldsymbol{\mu}_1$ would be trained with $\mathbf{x}_1$, and $\boldsymbol{\mu}_2$ would be trained with $\mathbf{x}_1$ and $\mathbf{x}_2$. In the modified MCE training for detectors $\mathbf{x}_2$ is not classified as c_1 (the target class) in any segmentation. Therefore, $\boldsymbol{\mu}_1$ is trained with $\mathbf{x}_1$ as before, but $\boldsymbol{\mu}_2$ is now trained only with $\mathbf{x}_1$.

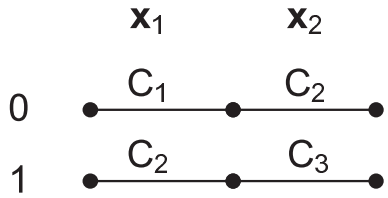


Figure 4.1: Example: two transcriptions and two frames.

The motivation for this modification is to focus the training effort in those frames that present confusability between class and anticlass while avoiding frames with high confusability within the anti-class. This is better understood considering the analysis in [73, Section 5.3.1], where it was stated that "the (string-based MCE) optimization will focus on the differences between correct and recognized strings: the incorrectly recognized phoneme models will be pushed away and the correct models will be pulled closer. Thus, implicitly, it seems that string-level learning focuses on improving phone accuracy". However, there are two important issues to consider. Firstly, our method probably weakens the connection between the optimization method and the optimality of the performance function J. Secondly, note that there is not a direct relationship between the optimality of J and the optimality of any of the detection performance criteria that were discussed in Section 3.3.

The steps followed in each iteration n of this detector training are described in Algorithm 4.2. The first step is the extraction of new features using the filterbank matrix computed in the previous iteration. After that an N-best list is generated using the new features and the set of model means computed in the previous iteration. The 1-best transcription L_{k1} is aligned with the correct transcription L_{k0} to compute the detection score, for example F-score or A_c, as described in Section 3.3. We can denote the detection scores for transcriptions L_{kj} as S_{kj} and then the scores for the 1-best transcription are referred to as S_{k1}. Note that the score computation is not a requirement of the detector training itself, but it can be used to find the best training iteration. The next step is to generate Viterbi state alignments for the N-best list and the correct transcription, that is to generate L_{kj}^n. Further, the gradients of J with respect to the means and the filterbank matrix are computed with Eqs. 4.24, 4.27 and 4.29. The final step is to apply Algorithm 4.1 to compute the filterbank matrix and the HMMs for the next iteration. This is repeated until the algorithm reaches the $n_{\max}$ iteration or some performance condition is fulfilled.

Algorithm 4.2 Modified MCE training for subword detectors.

for $n = 1 : n_{\max}$ **do**
  MFCC extraction: $\mathbf{H}^n \Rightarrow \mathcal{X}^n$
  N-best segmentation
  Evaluation: S_{k1}
  Viterbi state alignment: L_{kj}^n
  Eqs. 4.24, 4.27 and 4.29
  RPROP: $\mathbf{H}^{n+1}$ and $\boldsymbol{\mu}^{n+1}$
end for

4.3 Minimum Detection Error Training

In this section we present Minimum Detection Error (MDE) training, which is the application of the MPE framework [26, 75] to detection of subwords. MPE optimizes the parameters of a recognizer to improve the overall phone accuracy. Specifically, a performance function is built that estimates the expected phone accuracy on the training set as a function of the model parameters. However, the MPE framework is independent on the specific evaluation measure that is used in the performance function.

The key idea in MDE is the generalization of the MPE framework in order to optimize any measure. Specifically, in this thesis we use MDE to train subword detectors to optimize a detector evaluation measure, for example those presented in Section 3.3. Therefore, the performance function J in MDE estimates the expected detector score on the training set as a function of the detector structure parameters $\mathbf{\Lambda}$:

$$J(\mathbf{\Lambda}) = \frac{1}{K} \sum_k \bar{S}_k(\mathbf{\Lambda}) \ , \tag{4.30}$$

where $\bar{S}_k(\mathbf{\Lambda})$ is the estimated expected detection score of sentence k given the current set of parameters $\mathbf{\Lambda}$ and K is the total number of sentences. We estimate this expectation using the set of transcriptions $\{L_{kj}\}_{j=0}^N$:

$$\bar{S}_k(\mathbf{\Lambda}) = \sum_j P(L_{kj}|\mathbf{X}_k, \mathbf{\Lambda}) S_{kj} \ , \tag{4.31}$$

where S_{kj} is the detection score for transcription L_{kj}. Note that given L_{kj} and the reference L_{k0}, the score S_{kj} does not dependend of $\mathbf{\Lambda}$.

We need to compute the gradient of J with respect to the system parameters $\mathbf{\Lambda}$. In the following we discard the constant $\frac{1}{K}$ in the performance function for simplicity. Applying the chain rule of differential calculus:

$$\frac{\partial J(\mathbf{\Lambda})}{\partial \mathbf{\Lambda}} = \sum_{k,j} S_{kj} \frac{\partial P(L_{kj}|\mathbf{X}_k, \mathbf{\Lambda})}{\partial \mathbf{\Lambda}} \ . \tag{4.32}$$

The posterior probability $P(L_{kj}|\mathbf{X}_k, \mathbf{\Lambda})$ can be expressed in terms of likelihoods applying Bayes' theorem:

$$P(L_{kj}|\mathbf{X}_k, \mathbf{\Lambda}) = \frac{p(\mathbf{X}_k|L_{kj}, \mathbf{\Lambda})P(L_{kj})}{p(\mathbf{X}_k|\mathbf{\Lambda})} \tag{4.33}$$

$$= \frac{p(\mathbf{X}_k|L_{kj}, \mathbf{\Lambda})P(L_{kj})}{\sum_u p(\mathbf{X}_k|L_k^u, \mathbf{\Lambda})P(L_k^u)} \ . \tag{4.34}$$

In addition we can consider that

$$\frac{\partial p(\mathbf{X}_k|L_{kj}, \mathbf{\Lambda})}{\partial \mathbf{\Lambda}} = p(\mathbf{X}_k|L_{kj}, \mathbf{\Lambda}) \frac{\partial \log p(\mathbf{X}_k|L_{kj}, \mathbf{\Lambda})}{\partial \mathbf{\Lambda}} \tag{4.35}$$

$$= p(\mathbf{X}_k|L_{kj}, \mathbf{\Lambda}) \frac{\partial g_j(\mathbf{X}_k; \mathbf{\Lambda})}{\partial \mathbf{\Lambda}} \ , \tag{4.36}$$

where the first equality results from derivative of the logarithm and the second one is obtained simply applying our notation for the log-likelihood, that is $g_j(\mathbf{X}_k; \mathbf{\Lambda}) = \log p(\mathbf{X}_k|L_{kj}, \mathbf{\Lambda})$. In Section B.2 it is shown that

$$\frac{\partial J(\mathbf{\Lambda})}{\partial \mathbf{\Lambda}} = \sum_{k,j} (S_{kj} - \bar{S}_k(\mathbf{\Lambda})) P(L_{kj}|\mathbf{X}_k, \mathbf{\Lambda}) \frac{\partial g_j(\mathbf{X}_k; \mathbf{\Lambda})}{\partial \mathbf{\Lambda}} \ . \tag{4.37}$$

It is important to notice that there are two approximations in our approach. Firstly, recall that the average performance $\bar{S}_k(\mathbf{\Lambda})$ is an estimate given that from all possible string sequences we only consider the correct transcription and those with highest log-likelihood (given by the N-best list). Secondly, the normalization factor $p(\mathbf{X}_k|\mathbf{\Lambda})$ in the posterior probabilities is approximated as

$$p(\mathbf{X}_k|\mathbf{\Lambda}) \approx \sum_{j=0}^{N} p(\mathbf{X}_k|L_{kj}, \mathbf{\Lambda}) P(L_{kj}) \tag{4.38}$$

for the same reasons as above. However, in practice we found that these estimates were effective even when the number of considered hypotheses was

small, for example $N = 10$. Note that the proposed MDE framework could use lattices instead of N-best lists, something which is common in state-of-the-art implementations of MPE for large-vocabulary continuous speech recognition tasks.

In principle, an advantage of MDE over the MCE-based method presented in the previous section is that it can be used to optimize *directly* any chosen detection performance measure. Note that in the MCE-based method the performance function J is not directly related to any detection evaluation criterion and, therefore this method probably leads to suboptimal results. In addition, this also means that MDE is more flexible than the MCE-based method because it can optimize detectors focusing on the specific figure-or-merit of the application. For example, while recall is prioritized in detection-based ASR, other applications could prefer an optimized F-score.

The steps in iteration n in the MDE algorithm are shown in Algorithm 4.3. The main difference with respect to Algorithm 4.2, apart from the equations, is the fact that MDE requires the computation of the detection scores S_{kj} for all transcriptions L_{kj}. The first step is the extraction of new features using the filterbank matrix computed in the previous iteration. After that an N-best list is generated using the new features and the set of model means computed in the previous iteration. All transcriptions L_{kj} are aligned with the correct transcription L_{k0} and the scores S_{kj} are computed. As in the previous method, S_{k1} can be used to choose the best iteration. The next step is to generate Viterbi state alignments for the N-best list and the correct transcription, that is to generate L^n_{kj}. Further, the gradients of J with respect to the means and the filterbank matrix are computed with Eqs. 4.37, 4.10 and 4.12. It should be emphasized that the formulae are applied without the modifications of Section 4.2.1. The final step is to apply Algorithm 4.1 to compute the filterbank matrix and the HMMs for the next iteration. This is repeated until the algorithm reaches the $n_{\max}$ iteration or some performance condition is fulfilled.

4.4 Summary

In this chapter we focused on discriminative training methods for subword detectors. Maximum Likelihood training does usually not lead to optimal classification performance. This is because training data is often limited and also because of the mismatch between the real distribution and the models used in the classifier. By contrast, discriminative training methods estimate parameters in order to optimize a function that is directly related to

Algorithm 4.3 Minimum Detection Error training for subword detectors.

for $n = 1 : n_{\max}$ **do**
  MFCC extraction: $\mathbf{H}^n \Rightarrow \mathcal{X}$
  N-best segmentation
  Evaluation S_{kj}, $\forall k, j$.
  Viterbi state alignment: L_{kj}^n
  Eq. 4.37, 4.10 and 4.12
  RPROP: $\mathbf{H}^{n+1}$ and $\boldsymbol{\mu}^{n+1}$
end for

the performance of the classifier. Three commonly used methods are Maximum Mutual Information (MMI), Minimum Classification Error (MCE) and Minimum Phone Error (MPE). MCE builds a performance function that embeds the decision rule implemented by the classifier. In MPE a performance function is built that estimates the expected phone accuracy on the training set as a function of the model parameters.

After that, we presented our mathematical framework for discriminative training of the parameters in a system. We saw that there are two main families of optimization methods: Grow Transformation and gradient techniques. Specifically, in this work we use RPROP, a gradient optimization method, which adapts the learning rates heuristically. The training algorithms that we have considered require the computation of the gradient of the log-likelihood with respect to the parameters to optimize. In this thesis we want to optimize the HMM means and filterbank matrix in subword detectors and, therefore, we started by computing the gradients of the log-likelihood with respect to these parameters.

We proposed two methods for discriminative training of subword detectors. The first training technique is based on MCE. In this method the frames used to train a given parameter are a subset of the frames that the standard method would use to train the same parameter. This is because we exclude those frames that potentially will not improve discrimination between class and anti-class. The second training method is Minimum Detection Error, which adapts the MPE framework to train subword detectors. In this method any detector performance evaluation criterion, for example the class accuracy or F-score, can be optimized directly.

Chapter 5

Detection-Based ASR

Subword detectors are useful within a number of applications. In this thesis we have focus on applying discriminatively trained subword detectors in detection-based ASR (DBASR) systems. Therefore, this chapter introduces this speech recognition paradigm. The outline of the chapter is as follows. Firstly, we describe the motivation behind the DBASR paradigm in Section 5.1. After that, the bank of detectors and the linguistic merger are discussed, respectively, in Sections 5.2 and 5.3. Finally, in Section 5.4 we propose a DBASR systems structure based on the subword detectors presented in Chapter 3 and the discriminative training methods described in Chapter 4.

5.1 The Detection-Based ASR Paradigm

One of the new speech recognition paradigms that has been suggested is DBASR. In [1] it was argued that the standard approach to ASR, which was reviewed in Chapter 2, is data driven and does not use all available knowledge about speech or language. In addition, it is not straightforward how to use knowledge sources in the standard approach ("knowledge-ignorant modeling").

There are a number of elements in a standard ASR system that apply speech knowledge. Firstly, we discussed in Section 2.4 how the MFCC extractor modifies the short-term spectrum based on properties of human hearing. Secondly, the commonly used left-to-right HMM for speech units is motivated by the quasi-stationarity of speech and this is a consequence of how humans produce speech. Thirdly, modelling of speech basic units such as phones requires phonetic knowledge. Further, the language model is based on phonetic, lexical and syntactic information.

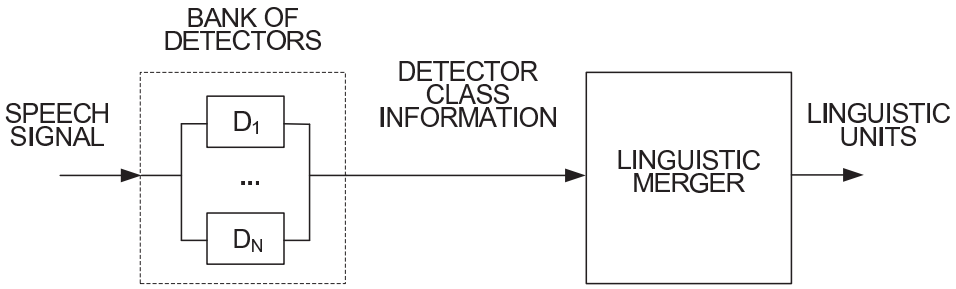


Figure 5.1: Block diagram of a DBASR system.

However, there are other knowledge sources that are not used in standard ASR systems. These speech attributes are mainly articulatory features of the speech signal, but they also include characteristics of the speaker (gender, accent, etc.) and the speaking environment. Recall that articulatory features provide information on the place (which articulator) and manner of production of the speech signal. A set of phones could then be represented by their values in the articulatory feature space. This would have the advantage that some phones with very different spectral properties would differ in few articulatory features, as discussed in [76]. For example /p/ and /b/ differ only in the presence of voicing.

A block diagram for the structure of a DBASR system is shown in Figure 5.1. The input to the system is the speech signal and the output is a hypothesized string of linguistic units, just as in a standard ASR system. The bank of detectors is built to find information about a set of intermediate classes. Specifically, each detector tries to find one of the intermediate classes in the speech signal and it outputs some kind of score for each frame or segment. The output from the bank of detectors is then processed by a linguistic merger and a sequence of linguistic units is hypothesized. In principle, the linguistic units in the output could be, for example, words, syllables or phones. However, in this thesis we have only considered mergers that output phones. If higher order units were required, they could be found, for example, with a dictionary and Viterbi decoding.

An important assumption in DBASR is that improving the detectors in the recognition system should lead to a better system performance. The detectors could be optimized by incorporating class specific knowledge. For example, vowel segments are known to be stable and thus longer windows could be used in the feature extractor of the detectors for vowel classes.

Another example is the use of acoustic parameters related the burst duration in order to discriminate stop sounds [77]. However, it is also possible to optimize the detectors using discriminative techniques.

5.2 Bank of Detectors

The structure of a DBASR system is flexible and allows a variety of detector alternatives. We have considered the detection framework described in Chapters 3 and the discriminative training methods presented in Chapter 4. In this section we discuss the choice of intermediate classes and, after that, some important issues in the output of the bank of detectors.

5.2.1 Intermediate Classes

A major issue when designing a DBASR system is how to choose the intermediate classes. The problem is close to the one of selecting appropriate basic speech units in a standard ASR system, as described in [53, p. 428]. The following subsections present some possible intermediate classes for a DBASR system: words, phones, syllables, acoustically derived subword units and articulatory features.

There are two points somehow related that should be considered. Firstly, it would probably be an advantage if there was a unique map between the intermediate classes and the linguistic units output by the merger. The detector structure that we have considered outputs class scores with continuous values and, therefore, it should be possible for the merger to separate linguistic units that share the same map to intermediate classes. However, having a unique map would probably offer more information redundancy and reliability. Secondly, in a DBASR system it is possible to combine different types of intermediate classes. For example, in [20] the bank of detectors had both articulatory feature and phone detectors. As in the previous case, more detectors in the system means in principle more reliability and redundancy. However, the task would probably be more complex for the linguistic merger. This is further discussed in Section 5.3.

Words

The chosen unit should be accurate, trainable and generalizable. Whole word models have the advantage that the inter-phonemic context dependence and coarticulation are incorporated in the model explicitly. However, in a general-purpose large-vocabulary ASR system whole-word models are not used as basic acoustic units. The reason is that the number of words is in

general too high to train even context-independent models. Therefore subword units are commonly used in large-vocabulary systems. For the same reasons, words are only suitable as intermediate classes in small-vocabulary DBASR systems.

Phones

A popular subword unit is the phone. The number of phones is much smaller than words and they all (at least monophones) occur so frequent that they are well trained. Phones are also generalizable because any word can be derived from phones by using a lexicon. An important practical advantage when using phonemes is the existence of a number of databases transcribed at the phonemic level. However context-independent phones are not very accurate because they overgeneralize the variability brought by coarticulation. If there is enough training data, a possible solution is to differentiate allophones according to their left and right neighbor phones (triphones).

Syllables

Syllables capture some coarticulation effects and they are more trainable than words. They can be a good compromise between words and phonemes in some languages such as Japanese, where there is small number of units. In other languages the number of syllables is, however, too high, for example over thirty thousand in English.

Acoustically Derived Subword Units

While the previous subword units were linguistically motivated, it is also possible to work with acoustically derived subword units (ASWUs) [78, 79]. These units are usually defined and found based on a pure acoustically-based clustering criterion. Therefore, if the detectors are based on HMM and acoustic features such MFCCs, ASWU should in principle be easier to detect than linguistically motivated units. During detector training transcriptions at the ASWU level are needed. However, ASWUs do not have a one to one mapping to linguistic units and, therefore, a lexicon is required. During the lexicon design phase the Viterbi algorithm is usually applied in order to find an ASWU pronunciation for each word realization. Thus, many different pronunciations are generated for the same word. In the final lexicon the pronunciation that is used could be, for example, the one with maximum average log-likelihood over all realizations of the word.

Articulatory Features

Articulatory features are speech features that are based on knowledge about human production and perception of speech. They are also referred to as attributes of the speech signal, distinctive features, phonological features or acoustic-phonetic features. Articulatory features have been investigated in connection with ASR, which was justified in [80] by the fact that "(1) such models should help account for coarticulation effects, (2) certain aspects of articulation can be more robustly detected than others, and (3) several classifiers, each with a small number of classes, may make better use of sparse training data than a single phone classifier".

Articulatory features have been used as intermediate classes in many DBASR systems [81]. There various articulatory feature sets based on different linguistic theories. For example, the Sound Pattern of English (SPE) [82] feature set and the multi-valued phonological feature set [83] are based on how and where the sounds are produced.

Very few databases are transcribed at the articulatory feature level and, therefore, it is common to work with a database transcribed at the phoneme level and a mapping that determines which articulatory features should be active for a given phoneme. There are, however, two points to consider. Firstly, it is important to notice that this is not a one-to-one mapping. Secondly, articulatory features can be asynchronized with the phoneme segmentations, they can be missing or even be present when they are not supposed to, for example because of coarticulation effects. Therefore, the resulting transcription is far from correct. Articulatory feature asynchrony and compensation methods were studied in [84].

5.2.2 Other Considerations

There are two important issues that appear when we consider the output of a bank of detectors. Firstly, more than one detector class can be active simultaneously or at least with some temporal overlap. For example, a given speech segment could be classified as the detection class by two different detectors. Intermediate classes can be exclusive and, therefore, we need a merger capable of considering the information from all the detectors to make an optimal decision. Secondly, the segmentations generated by different detectors will generally be asynchronized. This could be due to differences in window sizes in the feature extractors, segment durations, HMM parameters, etc. For example, in this thesis the segment-based structure presented in Section 3.2 is trained discriminatively for each detection class. The resulting models and filterbanks are different from each other

and, therefore, the segment borders and labels will in general be different as well.

In the classical "beads on a string" paradigm [85, 79] only one class can be active at a given time (frame) and there is no possibility of overlap. This paradigm is to some extent suitable to describe some intermediate classes, for example words, syllables or phones, but not articulatory features. This is because when humans produce speech the articulators achieve their targets asynchronously. In this sense, the possibility of overlap is an advantage of the detection-based paradigm. However, it also requires a complex merger capable of processing all the detector outputs and generating a single string of phonemes. The following section discusses a number of strategies to design the linguistic merger in a DBASR system.

5.3 Linguistic Merger

The role of the linguistic merger is to use the information from the bank of detectors to hypothesize a sequence of linguistic units, for example phones or words. Basically, the operation of the merger can be summarized in three steps. Firstly, the information from the bank of detectors is processed to generate a suitable input to the linguistic merger. Secondly, the input is mapped to scores for the chosen linguistic unit, which in our case is phones. Finally, an optimal sequence of linguistic units is found using the output scores. This can be done directly or with the generation of a lattice or N-best list as an intermediate step. The linguistic merger is commonly trained independently from the detectors, but training the detectors and the linguistic merger jointly can bring performance improvements [86, 87].

The structure of the merger is dependent on the choice of detectors. In this thesis we focus on segment-based detectors but, for completeness, we will also give a short overview of a merger choice for the frame-based detection case.

5.3.1 Merger for Frame-Based Detectors

In this case the output scores from each detector can be grouped together in vectors for every frame. Recall from Section 3.1.2 that frame-based detectors usually output posterior probabilities. If other scores are generated, for example likelihood, it is possible to process them to generate posterior scores. In addition, it may be necessary to normalize the detector scores.

If the intermediate classes are phones, a Viterbi decoder can be used to generate the optimal sequence of linguistic units. In the case where other

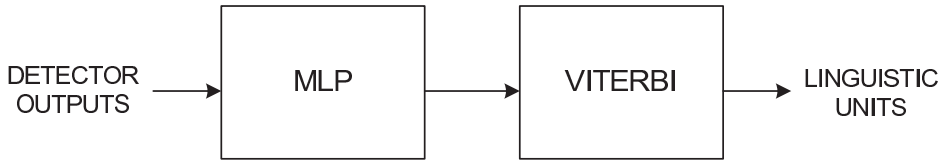


Figure 5.2: Block diagram of a hybrid MLP/Viterbi linguistic merger.

intermediate classes are chosen, for example articulatory features or ASWU, the intermediate class scores need to be mapped to phone scores before the Viterbi decoder.

Typically a Multi-Layer Perceptron (MLP) is used to process frame-based detector outputs. The detector scores (likelihood, posteriors or others) are normalized and mapped to phone or phone-state posteriors by the MLP. Further, a Viterbi decoder can be used to generate the best path. In this case, the merger results in the well known hybrid MLP/Viterbi [88]. Figure 5.2 shows a block diagram for this type of merger.

However, there are other possible merger structures. Conditional random fields were used to combine the output of MLP-based phone and articulatory feature detectors in [89]. Another possibility is to use HMMs with the detector scores as input features. For example, phone HMMs could be trained on the scores generated by a frame-based bank of detectors with articulatory features as intermediate classes, which would probably require a decorrelation stage as preprocessing. The final phone sequence would be decoded as in standard ASR systems, which was described in Section 2.2.4. In this case, the detector bank can be regarded as an advanced feature extractor followed by a standard ASR system. This method was followed in [76], where a triphone HMM-based large-vocabulary continuous speech recognizer was trained with knowledge-based features. The feature extractor in this case was a bank of articulatory feature detectors that were built with neural network and that had MFCCs as input features. The bank of detectors was followed by a Karhunen-Loeve transform in order to decorrelate the features.

5.3.2 Merger for Segment-Based Detectors

In the case of segment-based detectors, we have considered two possible strategies for the design of the linguistic merger. In the first method the merger works directly at the segment level. In this case, detector segment scores and segmentations are generated in the detector bank and then an

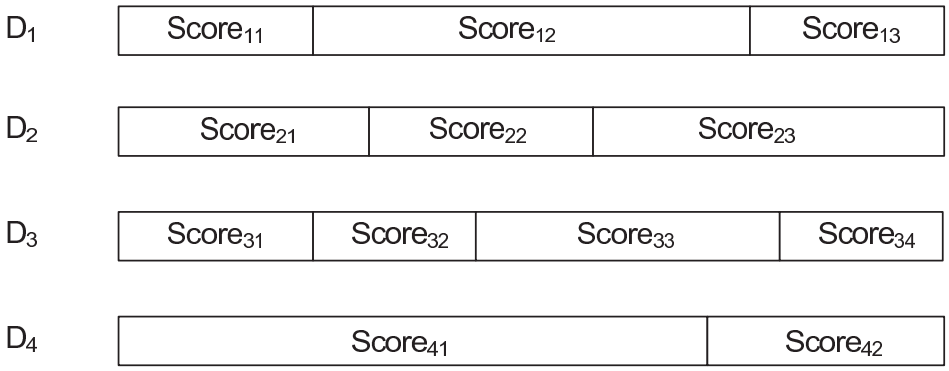


Figure 5.3: An example of output segmentation and scores.

algorithm finds the most likely phone sequence based on them. HMM-based detectors can generate a likelihood score for both the class and the anti-class models, which were described in Section 3.1.3. There are number of strategies if a single score is required for each segment. Firstly, it is possible to use only the class likelihood as segment score. Secondly, both likelihoods can be combined into a segment score through for example likelihood ratio, linear discriminant analysis, MLP, etc. Another possibility would be to use the segmentation output by the detector but to compute segment scores with a method independent of the class and anti-class models.

When the merger works directly at the segment level, the task of merging the information provided by the detectors is in general complex. This is because the bank of detectors can output asynchronous streams of information. Figure 5.3 shows an example of the segmentations and scores output by four detectors, where the segments start and end times are asynchronized and the number of segments is different. Merging asynchronous detector outputs is a difficult task and a research effort in this direction is still required. As far as we know, there are no experiments reported where the segment scores of a bank of HMM-based detectors have been directly merged into linguistic units. However, if the segmentation in the detectors is synchronized the task is simplified to some extent. In this case a possible strategy would be to group segment scores from all detectors into a vector and an MLP could generate phone posteriors. It would not be necessary to use a Viterbi decoder to generate a phone sequence because the MLP would already work at the segment level. Further, the synchronous segmentation in the detectors would not be modified by the MLP merger.

The second strategy for the linguistic merger is to generate scores at

the frame level from the detector segment scores. The resulting frame-based detector scores can be further processed as described in Section 5.3.1. For example, conditional random fields were used to merge HMM-based detectors in [14, 19, 20]. A simple method for deriving frame scores from the segment is to repeat the score of the segment, optionally normalized over the number of frames in the segment. Note that if the segmentation in the detectors is synchronized, this method will lead to identical frame-based vectors within a segment. Alternatively, in the case of HMM-based detectors Viterbi alignment could be used to find the frame-based state sequences and, therefore, a unique score for each detector state or frame could be generated. This method would increase the resolution because there would be less frames with the same score. However, even if the likelihood of the optimal state sequence is a good approximation to the total likelihood (Viterbi approximation), frame likelihoods obtained with this method show a big variation within a segment. Therefore, these frame scores could be smoothed by some method, for example median filtering. If the likelihood of both the class and anti-class are used to generate the segment score, for example with a likelihood ratio, we could use the state alignment of the anti-class as well. However, this would further increase the noisiness in the scores. In that case, a possible solution can be to use the state or frame likelihood for the class, and the segment likelihood for the anti-class. Other possible strategies were described in [16].

In addition, the resulting frame-based vectors could be augmented with time derivatives. In the case where all frames in a segment have been assigned an identical score there are two things to consider. Firstly, the derivatives should probably span several segment durations. Secondly, after including the derivatives, there would be fewer identical vectors because vectors corresponding to frames in the border would have different derivatives.

5.4 Proposed DBASR System

We propose a system where the detectors for the intermediate classes are built with the structure proposed in Section 3.2 and optimized with the subword detector training algorithms from Chapter 4, i.e. MCE-based or MDE training. In this thesis we have considered detectors for phones and articulatory features, but it would be possible to detect other intermediate classes, for examples syllables. Table 5.1 shows as an example four of the articulatory features (AFs) that we have considered and the phones where they are active. Appendix A describes in detail the phone and articulatory

Table 5.1: Examples of articulatory features

AF	Phones
Vocalic	aa, ae, ah, ao, eh, el, er, ey, ih, iy, l, ow, r, uh, uw, ux
Consonantal	b, bcl, d, dcl, dh, dx, el, em, en, eng, f, g, gcl, k, kcl, l, m, n, ng, nx, p, pcl, r, s, sh, t, tcl, th, v, z, zh
Nasal	em, en, eng, m, n, ng, nx
Low	aa, ae, ao, hh, hv, q

feature sets that we have considered and includes the complete mapping.

Figure 5.4 presents a block diagram of the proposed system. In this system each detector finds a segmentation for the speech signal and computes the likelihood of the class and anti-class models for each segment. The segment score is the log-likelihood ratio computed as

$$LLR = LL_c - LL_{ac} \ , \tag{5.1}$$

where LL_c and LL_{ac} are, respectively, the log-likelihood for the class and the anti-class models. Each frame is given the score of the corresponding segment, normalized by the number of frames in each segment. The frame scores from all detectors are grouped in a vector $\mathbf{u}$. Further, this vector $\mathbf{u}$ is augmented with first and second order derivatives, respectively $\mathbf{v}$ and $\mathbf{w}$. The first order derivatives are computed as

$$\mathbf{v}_t = \mathbf{u}_{t+L} - \mathbf{u}_{t-L} \ , \tag{5.2}$$

and similarly for the second order derivates. The parameter L is chosen so that the time derivative can include frames in other segments.

The linguistic merger consists on an MLP that generates phone posterior probabilities from the vector of detector scores appended with time derivatives. The final element is a Viterbi decoder that finds an optimal phone sequence.

5.5 Summary

This chapter introduced detection-based ASR (DBASR), one of the possible fields where subword detectors can be applied. This is an alternative paradigm for ASR where speech knowledge sources can be included in a

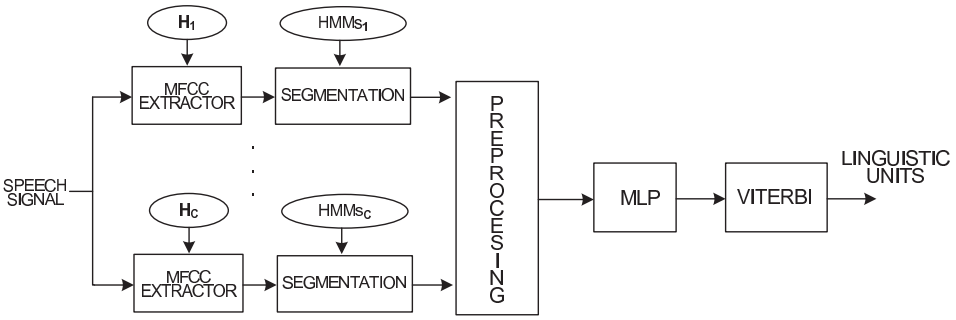


Figure 5.4: Proposed structure for detection-based ASR system.

system in the form of detectors. These knowledge sources are mainly articulatory features, which are related to the place (articulators) and manner in which the speech signal is produced. DBASR systems have two main parts: a bank of detectors for some intermediate classes and a linguistic merger. The most common intermediate classes are articulatory features but there are other possibilities, for example acoustically derived subword units, phones, syllables or words. The bank of detectors analyzes the speech signal and generate scores for the intermediate classes. The linguistic merger maps these scores to phone scores and finds an optimal sequence of phones. This is commonly done in three steps: preprocessing of the detector outputs, generation of linguistic unit scores and decoding. We have proposed a DBASR system structure where a bank of HMM-based detectors generate segment scores. Further, these scores are processed by a merger based on an MLP and a Viterbi decoder to generate a final phone hypothesis.

Part II

Experiments

Chapter 6

Experimental Settings

In this second part of the thesis we present experiments with the subword detectors that we have designed. Chapters 7 and 8 describe the experiments on optimization of subword detectors for phones and articulatory features using, respectively, the MCE and MDE training methods. Parts of this work were also presented in [47, 48]. In Chapter 9 we used the optimized detectors in detection-based automatic speech recognition (DBASR) systems.

Most of the experimental settings are common in those experiments and, therefore, we have chosen to present them in this chapter. Firstly, we describe the database that was chosen for the experiments. Secondly, we present the sets of intermediate classes that we considered. After that the acoustical parameterization is described. Finally, the chosen HMM structure is described, as well as the standard filterbank, discriminative training parameters and other important details on the experimental setup.

6.1 Database

Experiments were performed on the TIMIT acoustic-phonetic continuous speech corpus [90]. We chose this database because it is a well-known standard reference and it is labeled at the phoneme level. We used the designated training set of 462 speakers (3696 sentences), that is excluding SA-sentences. A development set of 50 speakers (400 sentences) was used for intermediate experiments. Results are reported on the NIST defined core test set of 24 speakers (192 sentences).

6.2 Phone and Articulatory Feature Sets

The experiments with phone detectors were based on a set of 39 phonemes. The manual TIMIT labelling consists of 61 acoustic-phonetic symbols. We merged plosive closures and bursts and applied the standard mapping to 39 phones [91].

The articulatory feature detection experiments were based on a set of 20 features similar to Sound Pattern of English (SPE), for example vocalic, strident, nasal, syllabic, etc. The articulatory features were derived by a mapping from a set of 56 phonemes where closures were preserved, /jh/ and /ch/ were mapped to /zh/ and /sh/ respectively, and the diphthongs were divided into a vowel and a glide. This is described in more detail in Appendix A.

6.3 Parameterization

The acoustic parameterization consisted of 13 static MFCCs (including C_0) with their first and second order derivatives. The sampling frequency was 16 kHz, and frames were extracted every 10 ms with 25 ms Hamming window. The filterbank was specific to each detector.

The standard filterbank had 26 triangular shaped filters where the center frequencies and bandwidths were uniformly spaced according to the Mel-scale (see Section 2.4 and Figure 7.1(a)). The total number of points in each filter corresponds to the number of points in the linear frequency spectrum, which was set to 201 points. This was a compromise between training complexity and performance.

6.4 Experimental Setup

The next two sections present the experimental setup for, respectively, the detection and recognition experiments. Figure 6.1 presents a block diagram that explains how the detectors and systems in our experiments are built for a given subword type (phones or articulatory features).

6.4.1 Detection Experiments

Baseline phone detectors were built using the standard filterbank and a set of 39 maximum likelihood trained HMMs. For the articulatory feature detectors we used a set of 56 HMMs. Figure 6.1 refers to the initial parameters as $\mathbf{\Lambda}_{\text{INI}}$. In each case, monophone HMMs were trained using the

phonemic transcription of TIMIT. The chosen HMM structure was 3 states in a left-to-right configuration, which is standard for phoneme modelling. The state output probability functions were GMMs with 10 mixtures and diagonal covariance matrices.

Two experiments were performed in the MCE and MDE training of subword detectors: F-score and class accuracy optimization. In each experiment, three implementations of the detector training were tested:

1. Only means were trained, referred to as (μ).
2. Only filterbanks were trained, referred to as (H).
3. Both means and filterbanks were trained, referred to as (H, μ).

We can refer to these experiments as *Score-Training(Implementation)*, for example *A-MDE(H)* denotes class accuracy optimization through MDE training of the filterbank matrix $\mathbf{H}$. Figure 6.1 shows how detectors are built for a given implementation of each detector training type. For example, for MDE training, the initial parameters $\mathbf{\Lambda}_{\mathrm{INI}}$ are trained with MDE for each detector. This leads to an optimized parameter set $\mathbf{\Lambda}_{\mathrm{MDE}}$ for each detection class. Each of these sets is then used to build an MDE-optimized detector.

Given the large number of detectors, we have chosen to present a weighted average detector performance. For the F-score we computed

$$\bar{F} = \sum_i (N_i \cdot F_i) / \sum_i N_i \tag{6.1}$$

and for the class accuracy

$$\bar{A}_c = \sum_i (N_i \cdot A_i) / \sum_i N_i \ , \tag{6.2}$$

where the index i refers to the class number and N_i is the number of class segments.

In the implementation of discriminative trainings the initial detector is the same as the corresponding baseline detector. In all cases, the number of competing hypotheses N was set to 10. In addition, in the MCE implementation η was set to 15. The gradient optimization method was RPROP and cross validation with the development set was used to select the best iteration according to the the performance measure to optimize, i.e. class accuracy or F-score.

The HTK Toolkit [92] was used for standard ML embedded training of HMMs, generation of N-best segmentations, forced alignments, string alignments, etc. Finally, the grammar was an unconstrained phone loop, the language model used under training and testing was a 0-gram (uniform model) and the insertion penalty was optimized on the development set.

6.4.2 Recognition Experiments

The banks of detectors in the DBASR systems were built with MDE-optimized subword detectors. The following eight configurations were tried:

1. Phone detectors, A-$MDE(H, \mu)$.
2. Phone detectors, A-$MDE(\mu)$.
3. Phone detectors, F-$MDE(H, \mu)$.
4. Phone detectors, F-$MDE(\mu)$.
5. Articulatory feature detectors, A-$MDE(H, \mu)$.
6. Articulatory feature detectors, A-$MDE(\mu)$.
7. Articulatory feature detectors, F-$MDE(H, \mu)$.
8. Articulatory feature detectors, F-$MDE(\mu)$.

The lowest path in Figure 6.1 shows how these systems are built (note that this figure does not make difference between phones and articulatory features).

To compare the performance of these optimized DBASR systems, a number of baseline systems were designed. Firstly, three standard ASR systems based on phone HMMs were built. These systems had the following configurations:

1. ML-trained HMMs and standard filterbank.
2. MCE-trained HMMs and MCE-trained filterbank (joint training).
3. MPE-trained HMMs and standard filterbank.

In the case of MPE optimization there was no performance improvement when the filterbank was trained jointly with the HMMs. Therefore, we chose the system where only the models were trained. Figure 6.1 refers to these systems as Baseline ASR ML, Baseline ASR MCE and Baseline ASR MDE.

Secondly, three baseline DBASR systems were built where the detectors were based on the ASR systems above. This means that in each baseline DBASR system all segmentations are identical. However, the score computation is specific to each detector, as described in Section 5.4. Figure 6.1 refers to these systems as Baseline DBASR ML, Baseline DBASR MCE and Baseline DBASR MDE.

The setup for the standard ASR systems was the same as described in the previous section: 39 monophone HMMs with the same structure, standard ML embedded training with HTK, ML models as starting point for discriminative training, phone loop as language model, etc.

In all DBASR systems the MLP had 200 hidden nodes and 39 output nodes. In early experiments we found that further increase in the number of hidden nodes did not improve the system performance. The number of inputs in the MLP was three times the number of detectors because time derivatives were added to the LLR score vectors from the detectors. The MLP was trained with backpropagation on a least square error criterion and the output nonlinearities were sigmoids. In the case of the baseline DBASR systems, the targets for the training were based on the forced alignment transcription generated with the same models as in the detectors. In the optimized DBASR systems the targets were based on the forced alignment transcription generated with the ML models. The final Viterbi alignment imposed only a minimum duration of three frames and the insertion penalty was optimized on the development set.

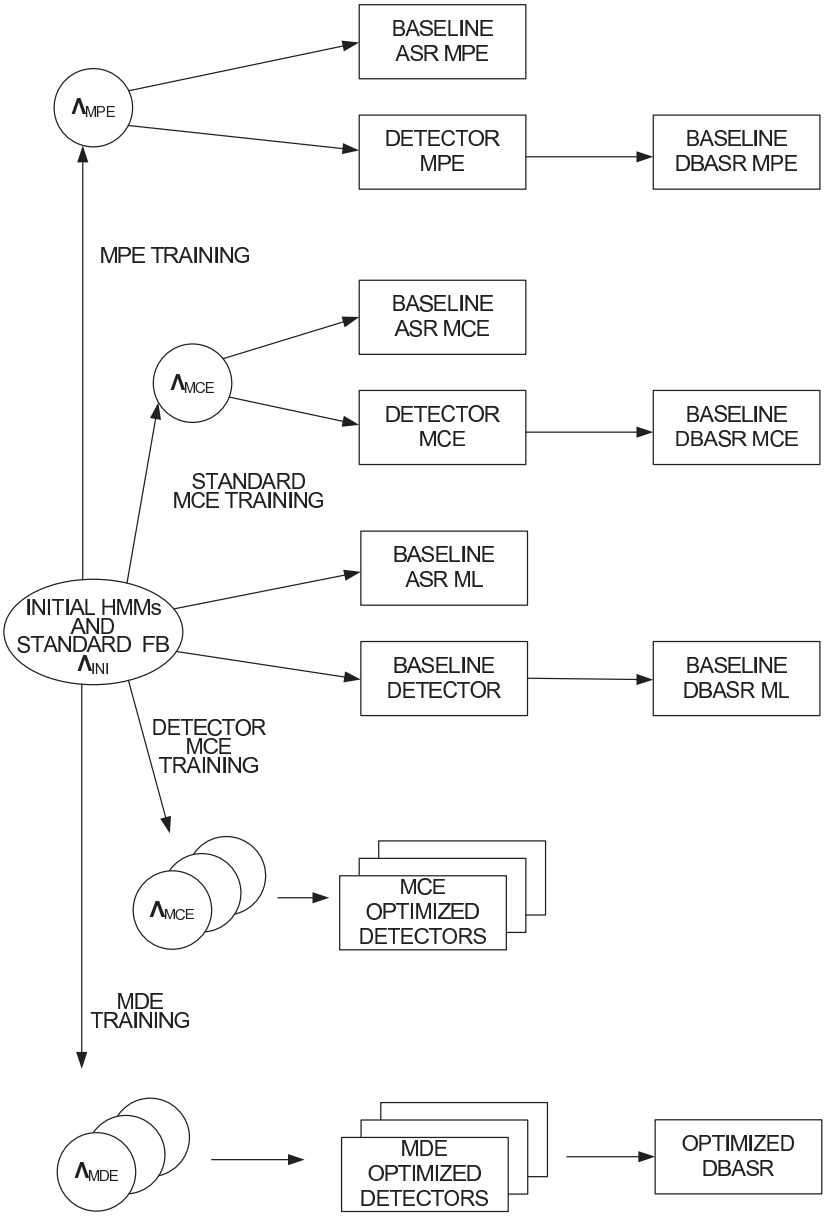


Figure 6.1: Block diagram of the experimental setup.

Chapter 7

Subword Detectors Trained with MCE

7.1 Task and Experimental Settings

In this chapter we present the experiments with subword detectors trained with the MCE training described in Section 4.2.1. Detectors with discriminatively trained filterbanks and models were applied to the task of phone and articulatory feature detection on TIMIT. Baseline detectors were built using the standard filterbank and a set of HMMs trained with maximum likelihood. Two MCE training experiments were performed in order to improve, respectively, the F-score and the class accuracy in the detectors. In each experiment three implementations of MCE were tested: $MCE(\mu)$, $MCE(H)$ and $MCE(H, \mu)$. The experimental setup was described in detail in Chapter 6.

7.2 Results and Discussion

This section presents the performance and filterbanks for the detectors. Firstly, we discuss the test results for the F-score and accuracy improvement experiments. They are presented in Tables 7.1 for phoneme detectors and 7.2 for articulatory feature detectors. The large number of detectors prevents us from presenting a thorough analysis of each case; instead we present a weighted average detector performance. In each experiment, we present the averages for both the F-score and the class accuracy of the detectors. However, we present also some specific examples of detectors: Table 7.3 shows the class accuracy, F-score, precision and recall for the

Table 7.1: Class averaged performance for the core test set. Phone detectors.

(a) F-score experiment.

Score	BL	*F-MCE(H)*	*F-MCE*(μ)	*F-MCE*(H, μ)
$\bar{F}$	67.0	66.5	69.2	69.3
$\bar{A}_c$	60.7	60.5	66.3	69.1

(b) Class accuracy experiment.

Score	BL	*A-MCE(H)*	*A-MCE*(μ)	*A-MCE*(H, μ)
$\bar{F}$	67.0	64.2	68.4	66.1
$\bar{A}_c$	60.7	62.3	67.0	73.7

detectors of /ih/, /n/ and /sh/. Secondly, we make a brief analysis of the trained filterbanks, focusing on those shown in Figures 7.1, 7.2 and 7.3. Figures 7.1(b) and 7.2(a) show only the filters in the low frequencies to display a better resolution.

As expected, discriminative training improved the performance of the detectors. In the class accuracy experiment, the relative error reductions with respect to baseline for phone detectors were 4.1% for *A-MCE(H)*, 16.0% for *A-MCE*(μ) and 33.1% for *A-MCE*(H, μ). For articulatory features, the corresponding class accuracy improvements were 3.7%, 30.9% and 43.1%. In the F-score experiment, the relative error reduction with respect to baseline for phone detectors were 6.7% for *F-MCE*(μ) and 7.0% for *F-MCE*(H, μ). For articulatory features, the corresponding F-score improvements were 9.0% and 9.0%. However, *F-MCE(H)* did not lead to improved performance neither for phone nor for articulatory feature detectors. This is discussed later in this section.

The average F-score in *F-MCE* experiments was higher than the corresponding F-score in *A-MCE* experiments, and vice versa. This is reasonable because increased recall usually comes at the expense of decreased precision. In addition, this shows that even if the training algorithm does not directly optimize class accuracy or F-score, both are improved during the training. In general the maximum F-score comes from a different training iteration than the maximum class-accuracy. However, for some detectors the best class accuracy and F-score were achieved in the same iteration, for example see Table 7.3(a) for *MCE*(μ).

In all cases, the average performance of *MCE*(μ) was higher than that

Table 7.2: Class averaged performance for the core test set. Articulatory Features.

(a) F-score experiment.

Score	BL	*F-MCE(H)*	*F-MCE*(μ)	*F-MCE*(H, μ)
$\bar{F}$	86.7	86.6	87.9	87.9
$\bar{A}_c$	81.2	81.3	85.8	85.8

(b) Class accuracy experiment.

Score	BL	*A-MCE(H)*	*A-MCE*(μ)	*A-MCE*(H, μ)
$\bar{F}$	86.7	86.1	87.7	85.9
$\bar{A}_c$	81.2	81.9	87.0	89.3

of *MCE*(H). For example, for phone detectors trained with *A-MCE*(μ) the relative error reduction with respect to *A-MCE*(H) was 12.5% . This can probably be explained by the fact that the total number of parameters in the HMM means was much higher than in the filterbank matrix (45630 vs. 5226). However, it is interesting to notice that in some detectors *A-MCE*(H) outperformed *A-MCE*(μ), for example Table 7.3(b) shows that for the /n/ detector the class accuracy was 64.2% vs. 62.6%.

For phone and articulatory feature detectors trained with *A-MCE*(H, μ) the relative error reduction with respect to *A-MCE*(μ) was, respectively, 20.3% and 17.7%. However, *F-MCE*(H, μ) did not outpeform *F-MCE*(μ) neither for phone nor for articulatory feature detectors. There are some possible explanations for the fact that *F-MCE*(H) did not improve the baseline performance and that *F-MCE*(H, μ) did not improve the performance of *F-MCE*(μ). Firstly, it could be that the standard filterbank is optimal for F-score. However, the results from the next chapter showed that this is not the case. In addition, note that even if the training did not improve the average performance, there were detectors where *MCE*(H, μ) did indeed improve the performance with respect to *MCE*(μ), for example Table 7.3(a) shows that for the /ih/ detector the F-score was 70.0% vs. 67.4%, and for the /n/ detector Table 7.3(b) shows 73.5% vs. 62.6%.

Secondly, recall that the MCE-based training described in Section 4.2.1 does not focus on optimizing the F-score directly and that, moreover, the number of parameters in the filterbank is lower than in the HMMs. Therefore, it is probably true that the filterbank parameters were not trained

Table 7.3: Performance of selected detectors after MCE-based detector training[1].

(a) /ih/

Training	A_c	F	P	R
BL	45.0	57.6	77.9	45.7
F-MCE(H)	38.4	51.8	76.7	39.1
A-MCE(H)	38.4	51.8	76.7	39.1
F-MCE(μ)	67.4	66.8	62.7	71.6
A-MCE(μ)	67.4	66.8	62.7	71.6
F-MCE(H, μ)	70.0	67.0	59.9	76.0
A-MCE(H, μ)	72.9	63.5	52.8	79.5

(b) /n/

Training	A_c	F	P	R
BL	57.7	69.5	84.7	59.0
F-MCE(H)	56.6	68.7	83.3	58.4
A-MCE(H)	64.2	61.6	57.7	66.0
F-MCE(μ)	62.6	69.9	72.1	67.8
A-MCE(μ)	62.6	69.9	72.1	67.8
F-MCE(H, μ)	73.5	72.1	67.1	77.9
A-MCE(H, μ)	73.5	69.6	62.3	79.0

(c) /sh/

Training	A_c	F	P	R
BL	76.6	76.4	75.0	77.9
F-MCE(H)	76.6	76.4	75.0	77.9
A-MCE(H)	76.6	76.4	75.0	77.9
F-MCE(μ)	74.0	78.2	77.2	79.2
A-MCE(μ)	74.0	78.2	77.2	79.2
F-MCE(H, μ)	70.1	79.1	88.7	71.4
A-MCE(H, μ)	85.7	66.7	52.6	90.9

[1]Class accuracy, F-score, precision and recall were defined in Section 3.3.

efficiently. Thirdly, in the case of *F-MCE*(H, μ), training of features and classifier simultaneously could increase the sensitivity of the parameters to the learning rates in a gradient-based optimization algorithm. In fact, training the complete filterbank matrix was found in [30] to be more sensitive to learning rates than other constrained configurations, for example training of amplitudes, centers and bandwidths. In addition, they also discussed that the conventional approach to avoid over-training may be inadequate for the case of simultaneous training of features and classifier. Another point to consider is the possibility that *F-MCE*(H, μ) gets stuck in a local minimum.

Some phone detectors for infrequent classes did not improve their performance in the development set after MCE-training, for example /ng/, /uh/ and /y/ with *A-MCE*(H, μ). We assume this was a problem of availability of training data. For those detectors, the initial (baseline) A_c was considered when computing $\bar{A}_c$. This is investigated in more detail in 9.2.

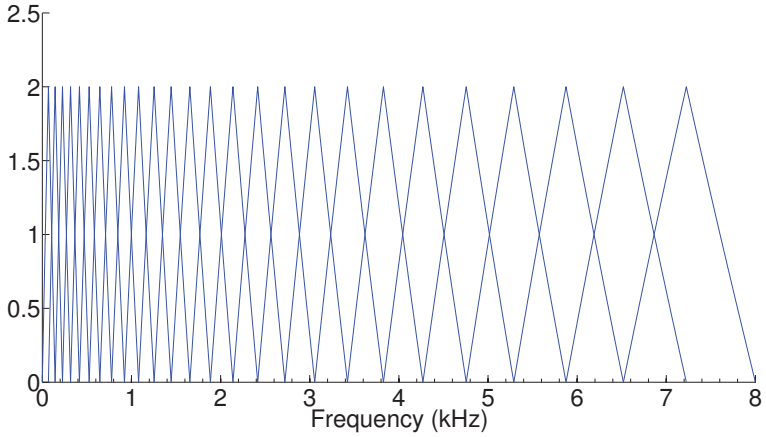
A significant part of the increase in performance brought by the new features can probably be explained by the fact that the filterbank in each detector was successfully modified to extract discriminative information for the specific detection task. Note that the filterbanks are clearly different; this is especially noticeable for classes with different acoustical properties, for example see filterbanks of the articulatory feature detectors for *vocalic* and *strident* in Figs. 7.2(b) and 7.3(b). Most of the changes in the filterbanks are due to scaling of the filter amplitudes and partly also different filter shapes. The filters were, however, only rarely shifted in frequency. We currently do not have a good explanation for this.

Analyzing the changes in the frequency form of the filterbanks resulted in some logical conclusions. We found that some of the significant changes often occurred at relevant frequencies, for example formant frequencies, for both the class and main competitors. In addition, the amplitude of a filter relative to the amplitudes of the surrounding filters seemed to carry discriminative information. Filterbanks in vowel detectors showed some similarities. Firstly, the main changes occurred in the lower frequencies $(0, 4)$ kHz, where the formants are located, and filters in the area (4-8kHz) were all attenuated. This is reasonable because it is known that formants are important for vowel classification. Secondly, in all vowel detectors the amplitude A_i of the first five filters followed 1) $A_1 < A_2 < A_3 < A_4 > A_5$, and 2) A_4 was the highest amplitude in the filterbank. This behavior in the lower filters is shown in the filterbanks of /ih/ and the articulatory feature *vocalic* (respectively in Figs. 7.2(a) and 7.2(b)) and it was probably related to the position of the first formant (F_1). Nasals have F_1 in the interval $(250, 300)$ Hz, which is lower than for vowels. This was reflected in a

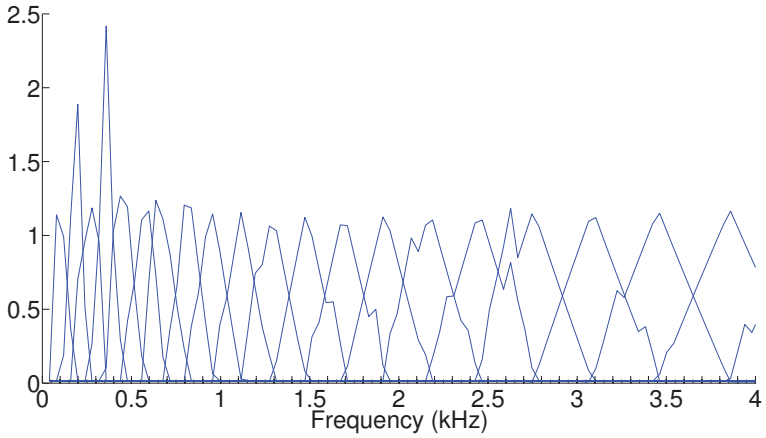
relatively higher A_2 than in vowel filterbanks, for example see the filterbank of /n/ in Fig. 7.1(b). In addition, it was noticeable that the vowels with lowest F_1, /iy/ and /uw/, had a relatively larger A_2 than other vowels. Filters in the higher frequencies in nasal filterbanks behaved as in the vowels, i.e. they were all attenuated. The filterbank for /sh/ is shown in Fig. 7.3(a). In this case the shape of the filters in the higher frequencies changed significantly, which did not happen for vowels and nasals. This agrees with the fact that high frequencies are important to discriminate fricatives. Moreover, /sh/ is one of the phones where the articulatory feature *strident* is present and Fig. 7.3(b) shows the same effect in the high frequencies.

7.3 Summary

In this chapter we built detectors of phones and articulatory features in continuous speech. These detectors had MFCC filterbanks and models improved with the detector MCE-training described in Section 4.2.1. For class accuracy improvement, phone detectors with MCE-trained filterbanks and HMMs reduced the average detector error rate by 33.1% compared to the baseline detectors and 20.3% compared to MCE-training of HMMs using standard MFCCs. For F-score improvement, phone detectors with MCE-trained HMMs reduced the average detector error rate by 6.3% compared to the baseline detectors, but MCE-training of filterbanks and HMMs did not lead to further improvements. Articulatory feature detectors showed similar results. In addition, we found that the trained filterbanks were clearly different and reflected acoustic properties, for example formant positions, of the class to detect.

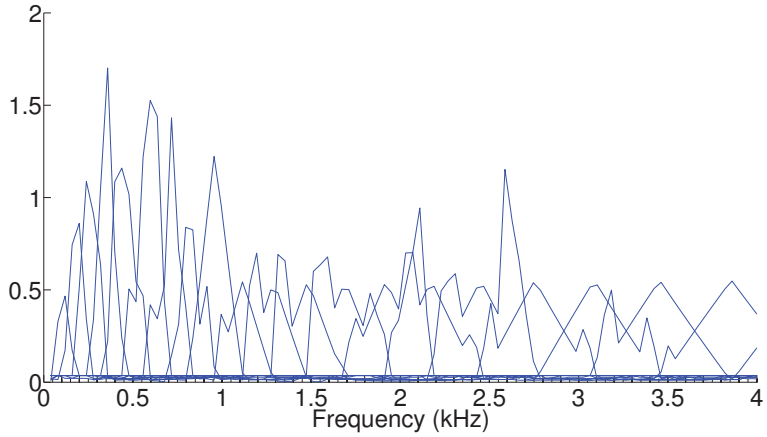


(a) Standard filterbank

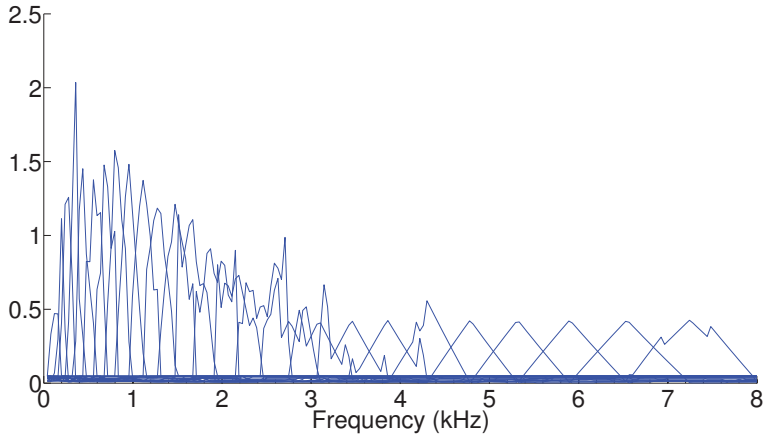


(b) /n/, A-MCE(H, μ)

Figure 7.1: Selected examples of filterbanks.

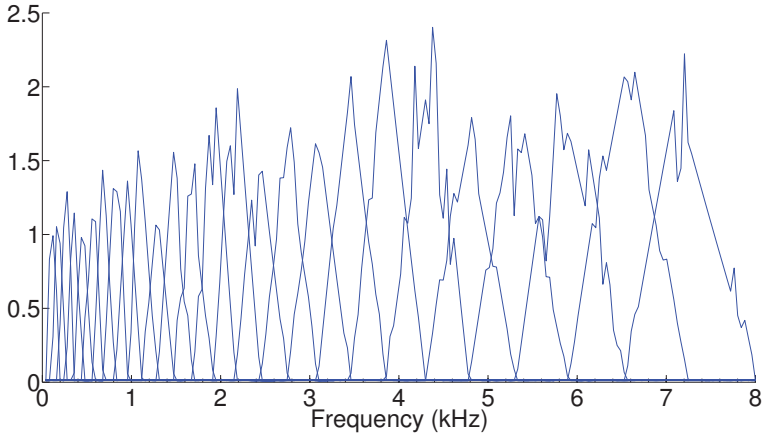


(a) /ih/, A-MCE(H, μ)

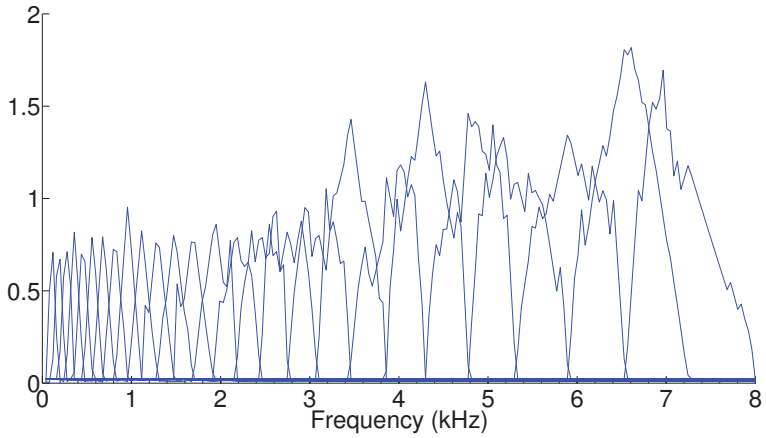


(b) VOC, A-MCE(H, μ)

Figure 7.2: Selected examples of filterbanks.



(a) /sh/, A-MCE(H, μ)



(b) STR, A-MCE(H, μ)

Figure 7.3: Selected examples of filterbanks.

Chapter 8

Subword Detectors Trained with MDE

8.1 Task and Experimental Settings

In this chapter we present the experiments with subword detectors optimized with the MDE training described in Section 4.3. Detectors with optimized filterbanks and models were applied to the task of phone and articulatory feature detection on TIMIT. Two MDE training experiments were performed: F-score and class accuracy optimization. In each experiment three implementations of MDE were tested: $MDE(\mu)$, $MDE(H)$ and $MDE(H, \mu)$. The experimental setup was described in detail in Chapter 6.

8.2 Results and Discussion

This section presents the performance and filterbanks for the detectors. Firstly, we discuss the test results for the F-score and accuracy optimization experiments. They are presented in Table 8.1 for phoneme detectors and Table 8.2 for articulatory feature detectors. As in the previous chapter, in each experiment we give both the averages for the F-score and the class accuracy. However, we present also some specific examples of detectors: Table 8.3 shows the class accuracy, F-score, precision and recall for the detectors of /ih/, /n/ and /sh/. We chose these detectors because they were used as specific examples in Chapter 7. Secondly, we make a brief analysis of the optimized filterbanks, focusing on those shown in Figures 8.2, 8.3 and 8.4.

As expected, discriminative training improved the average detector per-

Table 8.1: Class averaged performance for the core test set. Phoneme detectors.

(a) F-score experiment.

Score	BL	*F-MDE(H)*	*F-MDE*(μ)	*F-MDE*(H, μ)
$\bar{F}$	67.0	69.1	71.9	71.2
$\bar{A}_c$	60.7	63.9	66.6	66.3

(b) Class accuracy experiment.

Score	BL	*A-MDE(H)*	*A-MDE*(μ)	*A-MDE*(H, μ)
$\bar{F}$	67.0	67.2	69.9	67.5
$\bar{A}_c$	60.7	70.2	73.0	76.8

Table 8.2: Class averaged performance for the core test set. Articulatory Features.

(a) F-score experiment.

Score	BL	*F-MDE(H)*	*F-MDE*(μ)	*F-MDE*(H, μ)
$\bar{F}$	86.7	87.2	88.8	88.9
$\bar{A}_c$	81.2	82.9	85.4	86.0

(b) Class accuracy experiment.

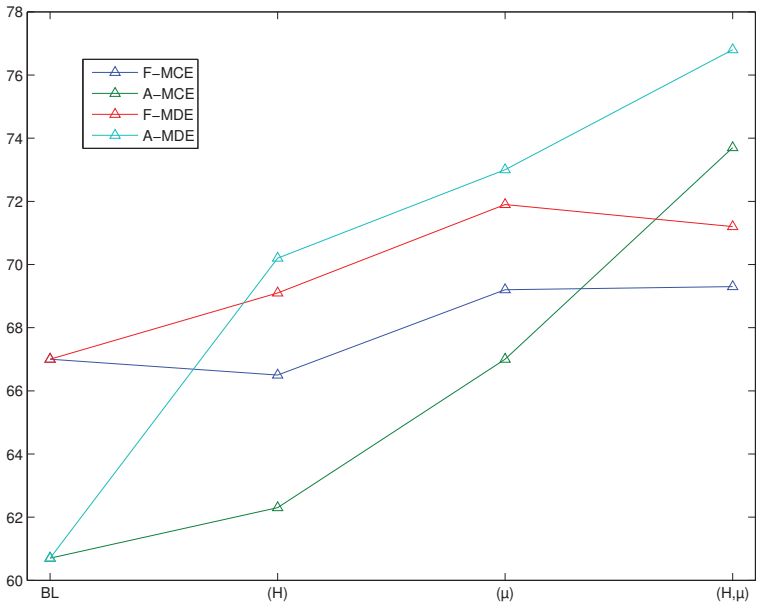
Score	BL	*A-MDE(H)*	*A-MDE*(μ)	*A-MDE*(H, μ)
$\bar{F}$	86.7	86.8	85.0	81.3
$\bar{A}_c$	81.2	87.8	93.1	95.1

formance in both experiments. In the F-score optimization experiment, the relative error reductions with respect to baseline for phone detectors were 6.4% for *F-MDE(H)*, 14.8% for *F-MDE*(μ) and 12.7% for *F-MDE*(H, μ). For articulatory features, the corresponding relative error reductions were 3.8%, 15.8% and 16.5%. In the class accuracy optimization experiment, the error reduction for phone detectors were 24.2% for *A-MDE(H)*, 31.3% for *A-MDE*(μ) and 41.0% for *A-MDE*(H, μ). For articulatory features, the corresponding relative error reductions were 35.1%, 63.3% and 73.9%.

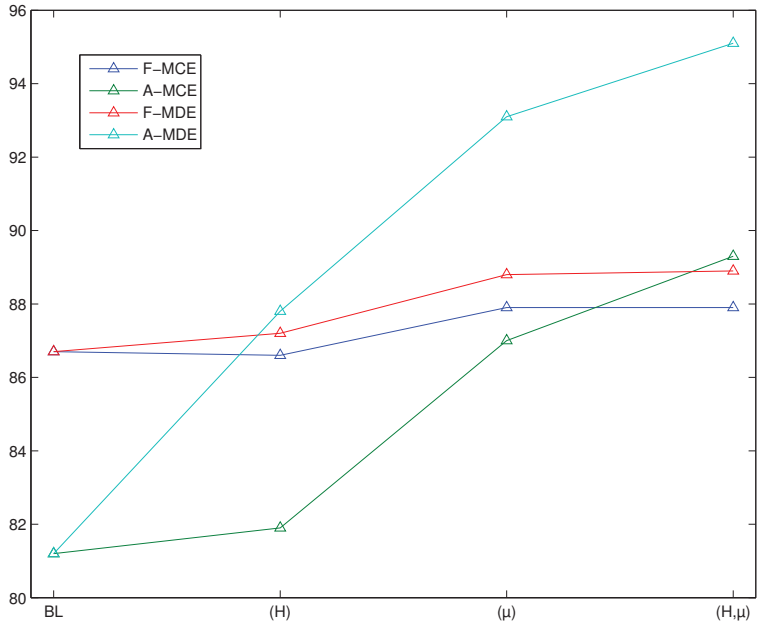
The average F-score in *F-MDE* experiments was higher than the corresponding F-score in *A-MDE* experiments, and vice versa. This is reasonable because increased recall usually comes at the expense of decreased precision. In addition, this shows that MDE-training focused on the optimization of the chosen performance criterion, F-score in the first experiment and class accuracy in the second. In both experiments, the average performance of *MDE*(μ) was higher than that of *MDE*(H). As it was mentioned in the previous chapter, this can probably be explained by the fact that the total number of parameters in the HMM means was much higher than in the filterbank matrix (45630 vs. 5226). However, in some detectors we found the opposite, e.g. Table 8.3(a) shows that *A-MDE*(H) improved the result of *A-MDE*(μ) for the /ih/ detector. Both for phone and articulatory feature detectors *A-MDE*(H, μ) reduced the error rate by, respectively, 14.1% and 29.0% with respect to *A-MDE*(μ). However, *F-MDE*(H, μ) did not improve the performance with respect to *F-MDE*(μ) neither for phone nor articulatory feature detectors. Some possible explanations were presented in the previous chapter.

We are also interested in comparing MDE training with the MCE-based detector training. Recall that, in contrast to the MCE-based method, MDE can optimize directly any detection evaluation criteria. Our approach in the MCE-based training was then to use the chosen performance metric in the cross validation, but this was at best suboptimal. The results in Tables 8.1, 8.2 and 8.3 can be compared, respectively, with those in Tables 7.1, 7.2 and 7.3. In order to simplify the analysis, Figure 8.1 shows the performance of MCE and MDE for phone and articulatory feature detectors. Note that only the average F-score performance is displayed for F-score experiments, and similarly for class accuracy experiments.

In the following we compare the performance of phone detectors trained with *A-MDE* and *A-MCE*. Firstly, Figure 8.1(a) shows that the performance curve for *A-MDE* was above the one of *A-MCE* in all cases. Note that the average performance of *A-MDE(H)* was also higher than that of *A-MCE*(μ) (70.2% vs 67.0%) even if the number of optimized parameters



(a) Phone detectors



(b) Articulatory feature detectors

Figure 8.1: Performance for different training methods: F-score for F-MCE and F-MDE, and accuracy for A-MCE and A-MDE.

was much smaller. In addition, in some cases detectors optimized with *A-MDE(H)* performed even better than those trained with *A-MCE*(H, μ), for instance comparing Table 8.3 and 7.3, we find 90.5% vs 72.9% for /ih/ and 74.8% vs 73.5% for /n/. Secondly, the average performance of *A-MDE*(μ) was close to *A-MCE*(H, μ) (73.0% vs 73.7%). Also in this case some detectors optimized with *A-MDE*(μ) performed better than those trained with *A-MCE*(H, μ), for example /ih/ and /n/ as well. Further, Figure 8.1(b) for articulatory feature detectors shows the same behavior. Note that in this case *A-MDE*(μ) was much higher than *A-MCE*(H, μ). Figures 8.1(a) and 8.1(b) show that the performance curve for *F-MDE* was also above that of *F-MCE* in all cases. In addition, it should be noted that while *F-MCE*(H) did not offer improvements with respect to baseline neither for phone nor for articulatory feature detectors, *F-MDE*(H) improved the baseline results in both cases and even offered performance similar to *F-MCE*(μ) for the case of phone detectors. Therefore, it can be concluded that that MDE training is more powerful than our previous MCE-based training for detectors.

Some phone detectors for infrequent classes did not improve their performance in the development set after MDE-training, specially with *MDE(H)*. For those detectors, the initial baseline score was considered when computing the average score. We assume this was a problem of training data availability. In addition, some detectors improved their performance with respect to baseline both for the training and development sets, while the corresponding test performances decreased, for example this was the case for the detector of /sh/ trained with *A-MDE*(μ). It is possible that this lack of generalization could be explained by the low number of class segments in the test set for the affected detectors. For example, the number of class segments in the training, development and test sets for /sh/ are, respectively, 1466, 153 and 77. This issues were also found in 7.2.

In the filterbank optimization experiments, *F-MDE(H)* and *A-MDE(H)*, the only changes in the detection structure with respect to baseline was the filterbank matrix $\mathbf{H}$. Therefore, the increase in performance brought by the new features can be explained by the fact that the filterbank in each detector was modified to extract discriminative information for the specific detection task. The filterbanks were clearly different from each other, specially for classes with different acoustical properties, for example see filterbanks for /ih/ and /sh/ in Figs. 8.3(b) and 8.4(b).

In the previous chapter we analyzed filterbanks optimized with *A-MCE*(H, μ). Most of the changes in the filterbanks were due to scaling of the filter amplitudes and partly also different filter shapes. However,

Table 8.3: Performance of selected detectors after MDE training[1].

(a) /ih/

Training	A_c	F	P	R
BL	45.0	57.6	77.9	45.7
F-MDE(H)	60.7	63.9	64.8	63.1
A-MDE(H)	90.5	47.6	31.5	97.8
F-MDE(μ)	64.1	69.0	70.3	67.8
A-MDE(μ)	84.1	62.1	47.4	89.8
F-MDE(H, μ)	66.0	67.9	65.4	70.7
A-MDE(H, μ)	91.7	56.3	39.9	95.9

(b) /n/

Training	A_c	F	P	R
BL	57.7	69.5	84.7	59.0
F-MDE(H)	70.4	73.6	73.7	73.5
A-MDE(H)	74.8	70.6	63.9	79.0
F-MDE(μ)	73.5	76.4	77.5	75.3
A-MDE(μ)	80.0	72.8	64.0	84.4
F-MDE(H, μ)	73.8	76.9	77.7	76.1
A-MDE(H, μ)	80.5	66.3	53.1	88.1

(c) /sh/

Training	A_c	F	P	R
BL	76.6	76.4	75.0	77.9
F-MDE(H)	68.8	77.5	84.6	71.4
A-MDE(H)	74.0	78.7	80.8	76.6
F-MDE(μ)	74.0	81.9	88.1	76.6
A-MDE(μ)	74.0	81.1	84.5	77.9
F-MDE(H, μ)	81.8	85.5	86.7	84.4
A-MDE(H, μ)	80.5	85.9	88.9	83.1

[1]Class accuracy, F-score, precision and recall were defined in Section 3.3.

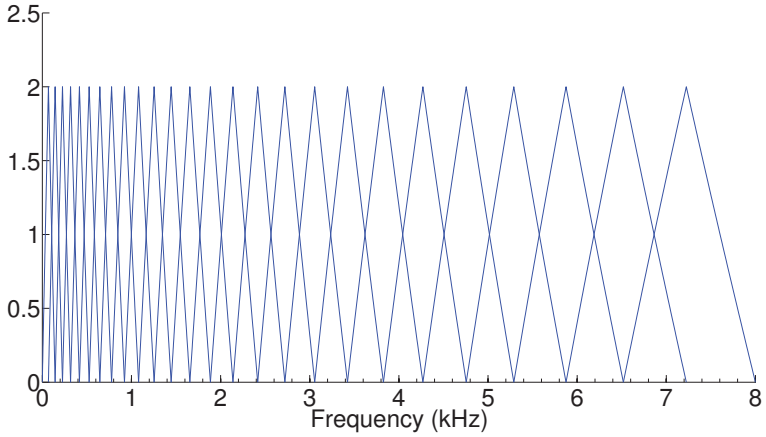
in *MDE(H)* detectors we found that optimized filters had often expanded in new frequencies. Therefore, the limitation in filterbank modifications found in the previous chapter are probably a consequence of the suboptimality of the MCE-based detector training. Since this cannot be visualized when all filterbanks are plotted together, Fig. 8.4(a) isolates the 18th filter from Fig. 8.3(b) as an example, and the initial filter is displayed as well as a reference. We can see 1) modified shape in the initial area $(2.8, 3.5)$ kHz, 2) smaller new values near 2.5 kHz, 3) significant new shape in the interval $(3.8, 4.4)$ kHz, with maximum near 4 kHz. This means that, in contrast to standard filters, this optimized filter outputs energy information from different critical bands.

Analyzing the changes in the frequency shape of the filterbanks resulted in some logical conclusions. The resulting filterbanks for the same detector in MDE(H) had some similarities in shape when optimizing for accuracy or F-score, for example compare Figs. 8.2(b) and 8.3(a). Some filterbanks reflected properties that were found in filterbanks optimized with A-MCE(H, μ), for example the filterbank in /n/ had a high amplitude in the second filter probably because nasals have a low first formant, see Figs. 8.2(b) and 8.3(a). In addition, some of the changes in vowels seemed to be related to the position of the formants as well. However, some of the properties that we found previously in A-MCE(H, μ) were not present in filterbanks optimized with *MDE(H)*, for example the shape of filters in the high frequencies in some vowel detectors was clearly different from the corresponding standard filters, see Fig. 8.3(b). These differences can probably be explained by 1) filterbanks were trained keeping the baseline models, while in our previous method both filterbanks and models were optimized and 2) MDE differs from MCE both with respect to algorithm and performance.

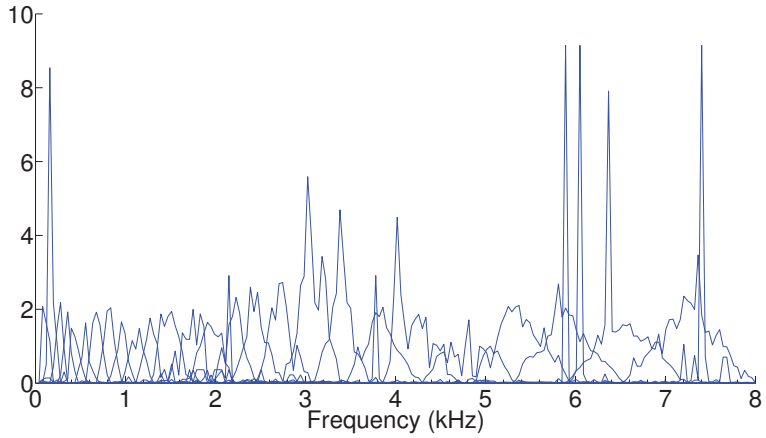
8.3 Summary

In this chapter we built phones and articulatory feature detectors where the MFCC filterbanks and HMMs were optimized with MDE training as described in Section 4.3. We found that our MDE technique succeeded in optimizing detectors for the chosen evaluation criteria; phone detectors optimized for F-score had a relative improvement of 14.8% over baseline, and the corresponding class accuracy improvement was 41.0%. Similar results were obtained for the articulatory feature detectors. The results showed that MDE training leads to significative improvements with respect to the training method based on MCE. Further, the optimized filterbanks reflected

acoustic properties of the detection class as we found in the previous chapter. However, the filterbanks were in this case significantly different than the standard filterbank. In fact some filters were modified to extract information from different critical bands.

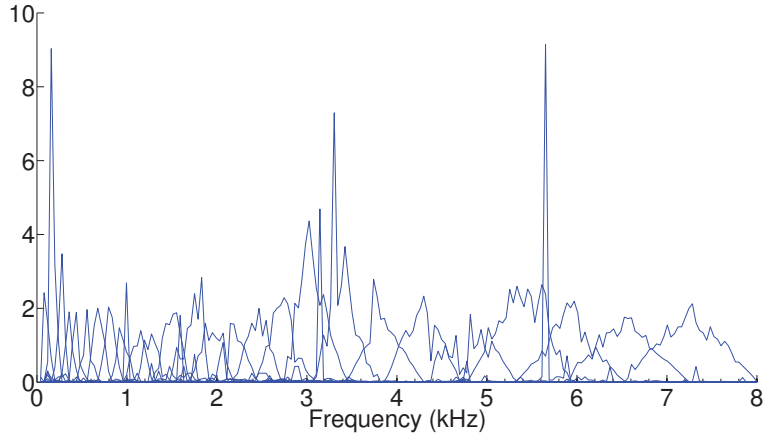


(a) Standard filterbank

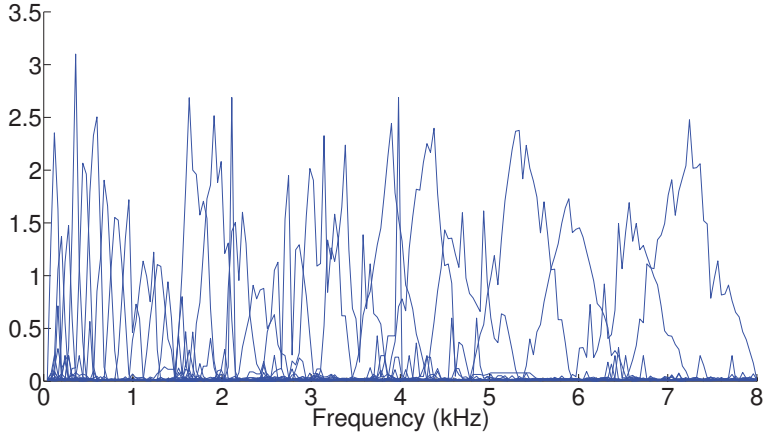


(b) /n/, *A-MDE(H)*

Figure 8.2: Selected examples of filterbanks.

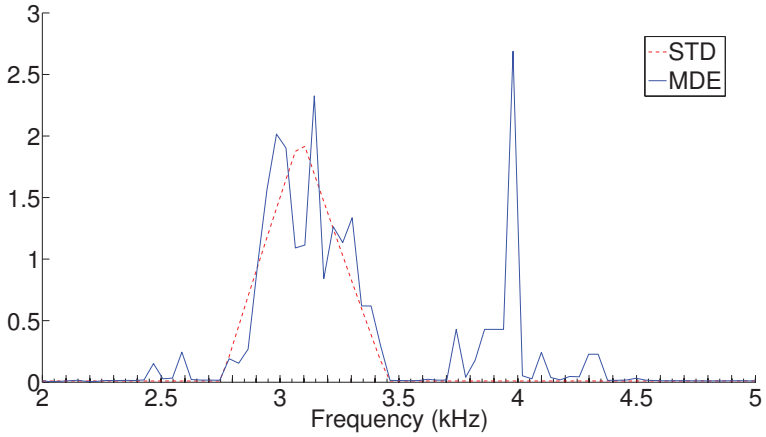


(a) /n/, *F-MDE(H)*

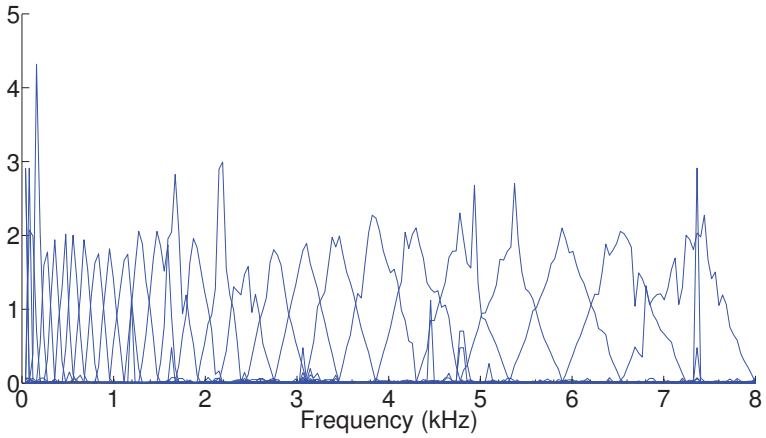


(b) /ih/, *F-MDE(H)*

Figure 8.3: Selected examples of filterbanks.



(a) /ih/, *F-MDE(H)*, filter 18



(b) /sh/, *A-MDE(H)*

Figure 8.4: Selected examples of filterbanks.

Chapter 9

Detection-Based ASR Experiments

9.1 Task and Experimental Settings

In this chapter we present experiments with detection-based automatic speech recognition (DBASR) systems built with the structure proposed in Section 5.4. These DBASR systems were applied to the task of phone recognition on TIMIT. The experimental setup was described in detail in Chapter 6.

9.2 Results and Discussion

This section presents the phone accuracy of the standard and detection-based ASR systems. Table 9.1 gives the performance of the standard (STD) and DBASR baseline systems. The performance for the optimized DBASR systems is given in Table 9.2, where IDC refers to the intermediate detection class in the bank of detectors. Firstly, the performance of the baseline systems and the DBASR systems is analyzed and discussed independently. After that, the performance of the optimized DBASR systems is compared to the baseline systems (STD and DBASR).

It should be noted that the phone accuracy of ML-trained standard ASR system is equal to the average class accuracy $\bar{A}_c$ of the baseline phone detectors (defined in Section 7.2), see for example Table 7.1(a). This can be explained as follows. Firstly, the total accuracy of a recognizer can be expressed as the average of the class accuracies (defined as in Eq. 3.5) weighted by the number of segments in each class, which equals $\bar{A}_c$. Secondly, recall

Table 9.1: Baseline systems: phone recognition accuracy for the test set.

System	ML	*MCE*(H, μ)	*MPE*(μ)
STD	60.7	62.7	66.1
DBASR	63.1	64.2	66.3

that the baseline detectors have the same structure as the baseline ML-trained ASR system.

Discriminative training of the ML-trained standard ASR systems led to performance improvements. The recognition results for the ML-trained standard ASR system and the relative improvement brought by standard embedded MCE are similar to those in [93, Table 2] (note their use of bigrams under testing). Further, as it could be expected the best performance for the standard baseline systems came from the MPE-trained system. This can be explained by the fact that the optimization criterion matches the system evaluation criterion, i.e. phone accuracy.

Baseline DBASR systems outperformed their corresponding standard ASR systems. However, in the case of the baseline DBASR system based on MPE-trained models, the performance gain is probably not significant. The baseline DBASR systems can be regarded as a rescoring scheme for the baseline standard ASR systems. The standard MPE-trained system is optimal in the sense that this rescoring scheme could not improve its performance.

The best DBASR system performance was given by the system built with articulatory feature (AFs) detectors trained with *F-MDE*(μ), with a 5.9% relative error reduction over the standard ML baseline system. The best performance for a DBASR system based on phone detectors brought a 2.8% relative error reduction over the standard ML baseline system. There are two possible explanations that would support the fact that articulatory feature detectors could lead to higher system performance. The first one is that the number of models in the detectors was higher (56 vs. 39). The second one is that since the number of detectors was smaller, the number of asynchronous streams was reduced and, therefore, the task was easier for the MLP. However, *A-MDE* phone detectors performed better than *A-MDE* articulatory feature detectors. We could not find an explanation for this behavior.

In all cases, *MDE*(μ) detectors led to a better system performance than the corresponding for *MDE*(H, μ) detectors. It should be noted that in all cases except F-MDE(μ) vs. F-MDE(H, μ), detectors with better detector

Table 9.2: DBASR systems: phone recognition accuracy for the test set.

IDC [1]	*A-MDE*(μ)	*A-MDE*(H, μ)	*F-MDE*(μ)	*F-MDE*(H, μ)
Phones	61.1	59.1	61.8	59.8
AFs	59.0	56.0	63.0	62.9

performance led to a worse system performance. For example *A-MDE*(μ) phone detectors had worse average class accuracy than *A-MDE*(H, μ), however *A-MDE*(μ) phone detectors led to a better system performance. A possible explanation is that detectors with class-specific feature extractors generate segmentations that are more asynchronized and, therefore, the merging task requires a more advanced merging structure than the MLP. Further, *F-MDE* detectors led to a better system performance than the corresponding for *A-MDE* detectors. This is very interesting because it has previously been stated that recall should be prioritized over precision for DBASR in order not to lose candidates, for example see [15]. A possible explanation is that recovering misses and prunning insertions are in fact equally difficult for the MLP.

The best DBASR system brought a phone recognition accuracy improvement with respect to the standard ML and MCE baselines (probably not significant in the latter case). However, the standard MPE and DBASR baselines were in all cases superior to optimized DBASR systems. Moreover, some of the optimized DBASR systems performed even worse than the standard ML baseline system. This can probably be explained by the fact that the proposed merger structure is too simple for the task of merging asynchronous detectors. This explanation is consistent with the fact that *MDE*(μ) detectors led to a better system performance than the corresponding for *MDE*(H, μ) detectors, even if *MDE*(μ) detectors had worse average detector performance. Note that the study on asynchrony compensation in [84] could be applied to our system in order to improve the results.

9.3 Summary

In this chapter we built detection-based automatic speech recognition systems for the task of phone recognition in TIMIT. The banks of detectors were built with subword detectors trained with MDE. The linguistic merger was based on a MLP and a Viterbi decoder. The best performance was achieved with a DBASR system built with F-MDE(μ) articulatory feature

[1] Intermediate detection class.

detectors, which brought a 5.9% relative error reduction over the standard ML baseline system. However, the performance of the best DBASR system was far from a standard baseline system trained with MPE. This can probably be explained by the fact that the linguistic merger is not capable of merging the asynchronous stream of information provided by the detectors.

Chapter 10

Conclusions and Future Work

In this final chapter we will summarize this thesis and present the most important conclusions from this work. After that we will give some suggestions for future work.

10.1 Conclusions

Firstly, we proposed a structure suitable for subword detection. This structure is based on the standard HMM framework, but in each detector the MFCC feature extractor and the models are trained for the specific detection problem. The experiments showed the effectiveness of this structure for detection of phones and articulatory features.

Two discriminative training techniques were proposed for detector training. The first one is a modification of Minimum Classification Error training. The second one, Minimum Detection Error training, is the adaptation of Minimum Phone Error to the detection problem. Both methods were used to train HMMs and filterbanks in the detectors, isolated or jointly. MDE has the advantage that any detection performance criterion can be optimized directly. F-score and class accuracy optimization experiments showed that MDE training is superior to the MCE-based method.

The optimized filterbanks reflected some acoustical properties of the detection classes. Moreover, some changes were consistent over classes with similar acoustical properties. In addition, MDE-training of filterbanks resulted in filters significatively different than in the standard filterbank. Some filters extracted information from different critical bands.

Finally, we proposed a detection-based automatic speech recognition system. Detectors are built with the proposed HMM-based detection structure and trained discriminatively. The linguistic merger is based on an MLP/Viterbi decoder. Experimental results showed that the improvements at the detection level brought by MDE training did lead to an increase in phone recognition accuracy at the system level. This can probably be explained by the fact that the proposed merger structure is not capable of merging the asynchronous information output by the detectors.

10.2 Future Work

In the following we give some suggestions for future research in this area. The proposed detection structure could be further improved for each detection class. Firstly, MFCC features could be tuned for the specific detection class. This could be done, for example, by using specific window sizes, frame steps, etc. Secondly, the model structure could also be optimized for each detector. For example, the number of states in the HMMs, number of mixtures in the GMM or even the state output functions could be specific to each detector.

For the optimization of detector-specific filterbanks, it would be interesting to study HMM-state specific filterbanks, which would model some of the short time dynamic of the signal that is relevant for the detection task. In addition, filterbanks based on phonetic knowledge of the target classes could be discriminatively trained and the results compared to those obtained in this work. The filterbanks obtained with MDE training could be analyzed thoroughly and compared to existing acoustic knowledge of phones and articulatory features.

In the experiments there were a number of issues that require further investigation. Firstly, in the class accuracy experiments in Chapters 7 and 8 we found that joint discriminative training of HMMs and the filterbank matrix led to improvements with respect to training the HMMs only. However, this was not the case in the F-score experiments. We discussed a number of reasons that could explain this behavior, for example that joint training increases the sensitivity to training parameters, that the conventional approach to avoid over-training may be inadequate for the case of simultaneous training of features and classifier, or that *F-MCE*(H, μ) gets stuck in a local minimum.

Secondly, in the experiments with DBASR systems in Chapter 9 we found that the system built with *F-MDE*(μ) articulatory feature detectors outperformed the DBASR system with phone detectors trained with the

same method. However, the system built with phone detectors trained with *F-MDE*(μ) outperformed the one built with articulatory feature detectors. We did not find a good explanation for this contradictory behavior. Further, bank of detectors with lower average performance at the intermediate class level led to better DBASR system performance. We explained this by limitations of the MLP-based merger, but this should probably be studied in more detail.

The detectors that we have developed could be used to build a pronunciation training system that focuses on vowel quality, plosive confusion, etc. Further, for the application to detection-based automatic speech recognition systems, a research effort is needed for the linguistic merger. The detectors that we have developed are asynchronous and the experiments in Chapter 9 showed that merging that information requires a more advanced structure than the proposed MLP/Viterbi. A possible direction is the use of conditional random fields.

Appendix A

Sets of Intermediate Classes

This appendix describes the phoneme and articulatory feature sets that were used for the experiments with the TIMIT acoustic-phonetic continuous speech corpus [90]. Firstly, a 39 phone set was derived as follows. The manual TIMIT labeling consists of 61 acoustic-phonetic symbols. We merged plosive closures and bursts, and /eng/ was mapped to /ng/ due to few training samples. This reduced the number of symbols in the standard set to 54. Further, the standard mapping to 39 phones was applied [91]. The resulting set is shown in Tables A.2 and A.3.

Secondly, a set of articulatory features was defined in the SIRKUS[1](Spoken Information Retrieval by Knowledge Utilization in Statistical speech processing) project in the speech group at the Department of Electronics and Telecommunications (NTNU). This project was related to the detection-based ASR (DBASR) paradigm and its objective was to improve the performance of ASR by integrating speech knowledge into a statistical framework. In this thesis we have discussed that in DBASR systems this has been usually accomplished by using articulatory features instead of or in addition to phones. Therefore, a set of 20 articulatory features (shown in Table A.1) was developed for DBASR experiments. The definition of these articulatory features was based on phonetics and did not considered the existence of methods to detect them automatically. All articulatory features in the set are strictly binary. Further, the set is based on the Sound Pattern of English (SPE) [82], but there are also elements from [94, 95]. However, the articulatory features were adapted to TIMIT because in this database there are several allophonic variants of the same phoneme.

Further, this set of articulatory features is specified for a set of 56 phonemes derived from the original 61 phone set in TIMIT. The mapping is

[1]http://www.iet.ntnu.no/projects/sirkus/

shown in Tables A.4 and A.5. This 56 phone set was derived as follows. In order to have acoustically homogeneous segments diphthongs were divided into a vowel and a glide. Further, in the affricates the frication part /jh/ and /ch/ were mapped, respectively, to /zh/ and /sh/. Note that plosive closures and bursts were not merged in this case. The resulting set is shown in Tables A.2 and A.3.

It is important to notice that even if the features are defined over a reduced phone set, the defined mapping is not one to one for all phonemes. For example /n/ and /nx/ are defined for the same features.

Finally, Table A.6 includes as a reference the command files used to obtain the 56 and 39 phone using HLEd. This is a tool included in HTK to transform labels in transcription files.

Table A.1: Articulatory features used in the SIRKUS project.

Core SPE	Added Features
vocalic	syllabic
consonantal	sonorant
nasal	mid
low	front
high	central
back	lateral
round	distributed
anterior	
coronal	
continuant	
strident	
tense	
voiced	

Table A.2: Phoneme mappings I

61	Class	56	54	39
aa	vowels	aa	aa	**ao**
ae	vowels	ae	ae	ae
ah	vowels	ah	ah	ah
ao	vowels	ao	ao	ao
aw	vowels	**aa w**	aw	aw
ax	vowels	ax	ax	**ah**
axh	vowels	axh	axh	**ah**
axr	vowels	axr	axr	**er**
ay	vowels	**aa y**	ay	ay
b	stop	b	b	b
bcl	stop closure	bcl	merged	merged
ch	affricates	sh	ch	ch
d	stop	d	d	d
dcl	stop closure	dcl	merged	merged
dh	fricatives	dh	dh	dh
dx	stop	dx	dx	dx
eh	vowels	eh	eh	eh
el	vowels	el	el	**l**
em	nasals	em	em	**m**
en	nasals	en	en	**n**
eng	nasals	eng	**ng**	**ng**
epi	others	epi	epi	**sil**
er	vowels	er	er	er
ey	vowels	ey	ey	ey
f	fricatives	f	f	f
g	stop	g	g	g
gcl	stop closure	gcl	merged	merged
h#	others	sil	sil	sil
hh	semivowels and glides	hh	hh	hh
hv	semivowels and glides	hv	hv	**hh**
ih	vowels	ih	ih	ih

Table A.3: Phoneme mappings II

61	Class	56	54	39
ix	vowels	ix	ix	**ih**
iy	vowels	iy	iy	iy
jh	affricates	zh	jh	jh
k	stop	k	k	k
kcl	stop closure	kcl	merged	merged
l	semivowels and glides	l	l	l
m	nasals	m	m	m
n	nasals	n	n	n
ng	nasals	ng	ng	ng
nx	nasals	nx	nx	**n**
ow	vowels	ow	ow	ow
oy	vowels	**ao y**	oy	oy
p	stop	p	p	p
pau	others	pau	pau	**sil**
pcl	stop closure	pcl	merged	merged
q	stop	q	q	removed
r	semivowels and glides	r	r	r
s	fricatives	s	s	s
sh	fricatives	sh	sh	sh
t	stop	t	t	t
tcl	stop closure	tcl	merged	merged
th	fricatives	th	th	th
uh	vowels	uh	uh	uh
uw	vowels	uw	uw	uw
ux	vowels	ux	ux	**uw**
v	fricatives	v	v	v
w	semivowels and glides	w	w	w
y	semivowels and glides	y	y	y
z	fricatives	z	z	z
zh	fricatives	zh	zh	**sh**

Table A.4: Articulatory feature mappings I

	SPE defined features													Added features						
Phone	VOC	CNS	NAS	LOW	HIG	BAC	ROU	ANT	COR	CNT	STR	TEN	VOI	SYL	SON	MID	FRO	CEN	LAT	DIS
sil	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
aa	1	0	0	1	0	1	0	0	0	1	0	1	1	1	1	0	0	0	0	0
ae	1	0	0	1	0	0	0	0	0	1	0	0	1	1	1	0	1	0	0	0
ah	1	0	0	0	0	1	0	0	0	1	0	0	1	1	1	1	0	0	0	0
ao	1	0	0	1	0	1	1	0	0	1	0	0	1	1	1	0	0	0	0	0
ax	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	1	0	1	0	0
axh	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	0	1	0	0
axr	0	0	0	0	0	0	1	0	1	1	0	0	1	1	1	1	0	1	0	0
b	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	1
bcl	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	1
d	0	1	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0
dcl	0	1	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0
dh	0	1	0	0	0	0	0	1	1	1	0	0	1	0	0	0	0	0	0	1
dx	0	1	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0
eh	1	0	0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	0	0	0
el	1	1	0	0	0	0	0	1	1	1	0	0	1	1	1	0	0	0	1	0
em	0	1	1	0	0	0	0	1	0	0	0	0	1	1	1	0	0	0	0	1
en	0	1	1	0	0	0	0	1	1	0	0	0	1	1	1	0	0	0	0	0
eng	0	1	1	0	1	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0
epi	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
er	1	0	0	0	0	0	1	0	1	1	0	1	1	1	1	1	0	1	0	0
ey	1	0	0	0	0	0	0	0	0	1	0	1	1	1	1	1	1	0	0	0
f	0	1	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	1
g	0	1	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
gcl	0	1	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
hh	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
hv	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0
ih	1	0	0	0	1	0	0	0	0	1	0	0	1	1	1	0	1	0	0	0

Table A.5: Articulatory feature mappings II

	SPE defined features													Added features						
Phone	VOC	CNS	NAS	LOW	HIG	BAC	ROU	ANT	COR	CNT	STR	TEN	VOI	SYL	SON	MID	FRO	CEN	LAT	DIS
ix	0	0	0	0	1	0	0	0	0	1	0	0	1	1	1	0	1	1	0	0
iy	1	0	0	0	1	0	0	0	0	1	0	1	1	1	1	0	1	0	0	0
k	0	1	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0
kcl	0	1	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0
l	1	1	0	0	0	0	0	1	1	1	0	0	1	0	1	0	0	0	1	0
m	0	1	1	0	0	0	0	1	0	0	0	0	1	0	1	0	0	0	0	1
n	0	1	1	0	0	0	0	1	1	0	0	0	1	0	1	0	0	0	0	0
ng	0	1	1	0	1	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0
nx	0	1	1	0	0	0	0	1	1	0	0	0	1	0	1	0	0	0	0	0
ow	1	0	0	0	0	1	1	0	0	1	0	1	1	1	1	1	0	0	0	0
p	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1
pau	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
pcl	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1
q	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
r	1	1	0	0	0	0	0	0	1	1	0	0	1	0	1	0	0	0	0	0
s	0	1	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0
sh	0	1	0	0	1	0	0	0	1	1	1	1	0	0	0	0	0	0	0	1
t	0	1	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0
tcl	0	1	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0
th	0	1	0	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0	1
uh	1	0	0	0	1	1	1	0	0	1	0	0	1	1	1	0	0	0	0	0
uw	1	0	0	0	1	1	1	0	0	1	0	1	1	1	1	0	0	0	0	0
ux	1	0	0	0	1	1	1	0	0	1	0	1	1	1	1	0	0	1	0	0
v	0	1	0	0	0	0	0	1	0	1	1	0	1	0	0	0	0	0	0	0
w	0	0	0	0	1	1	1	0	0	1	0	0	1	0	1	0	0	0	0	0
y	0	0	0	0	1	0	0	0	0	1	0	0	1	0	1	0	1	0	0	0
z	0	1	0	0	0	0	0	1	1	1	1	0	1	0	0	0	0	0	0	0
zh	0	1	0	0	1	0	0	0	1	1	1	0	1	0	0	0	0	0	0	1

Table A.6: HLEd Command files

61 to 56	61 to 54	54 to 39
RE aa_w aw RE aa_y ay RE ao_y oy EX RE zh jh RE sh ch RE axh ax-h RE sil h# SO	ME p pcl p ME t tcl t ME ch tcl ch ME k kcl k ME b bcl b ME d dcl d ME jh dcl jh ME g gcl g RE p pcl RE t tcl RE k kcl RE b bcl RE d dcl RE g gcl RE axh ax-h RE ng eng RE sil h# SO	DE q RE ax axh RE m em RE n nx RE hh hv RE uw ux RE er axr RE sil pau RE sh zh RE n en RE l el RE ao aa RE ah ax RE ih ix RE sil epi SO

Appendix B

Proofs of Results

B.1 Derivatives with Respect to the Filterbank Matrix

In this section we would like to find an expression for $\frac{\partial \mathbf{x}_t}{\partial \mathbf{H}} \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu} - \mathbf{x}_t)$. We use the notation in Figure 2.3. Each element $\frac{\partial \mathbf{x}_t}{\partial h_{kl}}$ is a row vector, which multiplied by $\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu} - \mathbf{x}_t)$ yields a scalar. We could divide $\mathbf{x}_t$, $\boldsymbol{\Sigma}^{-1}$ and $\boldsymbol{\mu}$ into static cepstrum, first derivative and acceleration parts:

$$\mathbf{x}_t = \begin{bmatrix} \mathbf{c}_t \\ \mathbf{d}_t \\ \mathbf{a}_t \end{bmatrix} \qquad \boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_c \\ \boldsymbol{\mu}_d \\ \boldsymbol{\mu}_a \end{bmatrix} \qquad \boldsymbol{\Sigma}^{-1} = \begin{bmatrix} \boldsymbol{\Sigma}_c^{-1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_d^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\Sigma}_a^{-1} \end{bmatrix} . \tag{B.1}$$

Then we have that

$$\begin{aligned} \frac{\partial \mathbf{x}_t}{\partial \mathbf{H}} \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu} - \mathbf{x}_t) = \frac{\partial \mathbf{c}_t}{\partial \mathbf{H}} \boldsymbol{\Sigma}_c^{-1}(\boldsymbol{\mu}_c - \mathbf{c}_t) + \frac{\partial \mathbf{d}_t}{\partial \mathbf{H}} \boldsymbol{\Sigma}_d^{-1}(\boldsymbol{\mu}_d - \mathbf{d}_t) + \\ \frac{\partial \mathbf{a}_t}{\partial \mathbf{H}} \boldsymbol{\Sigma}_a^{-1}(\boldsymbol{\mu}_a - \mathbf{a}_t) , \end{aligned} \tag{B.2}$$

where we have omitted the indices for model, state and mixtures for clarity.

We can start with the computation of the first element in Eq. B.2. Recall that $\mathbf{c}_t = \mathbf{D} \ln(\mathbf{y}_t)$, where $\mathbf{y}_t$ is a vector of N_{CH} components and that the DCT matrix has dimensions $N_{\text{CEP}} \times N_{\text{CH}}$. Applying the chain rule of differential calculus:

$$\frac{\partial \mathbf{c}_t}{\partial \mathbf{H}} \boldsymbol{\Sigma}_c^{-1}(\boldsymbol{\mu}_c - \mathbf{c}_t) = \frac{\partial \mathbf{y}_t}{\partial \mathbf{H}} ./ \mathbf{y}_t^T \mathbf{D}^T \boldsymbol{\Sigma}_c^{-1}(\boldsymbol{\mu}_c - \mathbf{c}_t) \tag{B.3}$$

$$= \frac{\partial \mathbf{y}_t}{\partial \mathbf{H}} (\mathbf{w}_t^c ./ \mathbf{y}_t) \tag{B.4}$$

where in the second equality we have defined $\mathbf{w}_t^c = \mathbf{D}^T \boldsymbol{\Sigma}_c^{-1}(\boldsymbol{\mu}_c - \mathbf{c}_t)$ and used the fact that $(\mathbf{u}./\mathbf{v})^T \mathbf{w} = \mathbf{u}^T(\mathbf{w}./\mathbf{v})$ holds for any two arbitrary vectors $\mathbf{u}$ and $\mathbf{v}$.

Further, considering that $\mathbf{y}_t = \mathbf{H}\mathbf{z}_t$, where the size of $\mathbf{z}_t$ is half of the spectrum given the symmetry properties of the FFT of a real signal, we obtain the following expression for each element in $\frac{\partial \mathbf{y}_t}{\partial \mathbf{H}}$:

$$\frac{\partial \mathbf{y}_t}{\partial h_{ij}} = z_{jt} \mathbf{u}_i^T \ , \tag{B.5}$$

where the vector $\mathbf{u}_i$ is a unit vector with a one in position i. Then it can be verified that

$$\frac{\partial \mathbf{x}_t}{\partial \mathbf{H}} \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu} - \mathbf{x}_t) = \frac{\partial \mathbf{y}_t}{\partial \mathbf{H}} (\mathbf{w}_t^c ./ \mathbf{y}_t) = (\mathbf{w}_t^c ./ \mathbf{y}_t) \mathbf{z}_t^T \ , \tag{B.6}$$

The computation of the other elements in Eq. B.2 requires an expression for the first and second order time derivatives as a function of the cepstral vector $\mathbf{c}_t$. Then we would have a number of terms of the form

$$\frac{\partial \mathbf{c}_{t+L}}{\partial \mathbf{H}} \boldsymbol{\Sigma}_d^{-1}(\boldsymbol{\mu}_d - \mathbf{d}_t)$$

and

$$\frac{\partial \mathbf{c}_{t+L}}{\partial \mathbf{H}} \boldsymbol{\Sigma}_a^{-1}(\boldsymbol{\mu}_a - \mathbf{a}_t) \ ,$$

where L is a positive or negative integer. Following same procedure as described for $\frac{\partial \mathbf{c}_t}{\partial \mathbf{H}} \boldsymbol{\Sigma}_c^{-1}(\boldsymbol{\mu}_c - \mathbf{c}_t)$ would lead to the following expressions:

$$\frac{\partial \mathbf{c}_{t+L}}{\partial \mathbf{H}} \boldsymbol{\Sigma}_d^{-1}(\boldsymbol{\mu}_d - \mathbf{d}_t) = (\mathbf{w}_t^d ./ \mathbf{y}_{t+L}) \mathbf{z}_{t+L}^T \tag{B.7}$$

and

$$\frac{\partial \mathbf{c}_{t+L}}{\partial \mathbf{H}} \boldsymbol{\Sigma}_a^{-1}(\boldsymbol{\mu}_a - \mathbf{a}_t) = (\mathbf{w}_t^a ./ \mathbf{y}_{t+L}) \mathbf{z}_{t+L}^T \ , \tag{B.8}$$

where we have defined $\mathbf{w}_t^d = \mathbf{D}^T \boldsymbol{\Sigma}_d^{-1}(\boldsymbol{\mu}_d - \mathbf{d}_t)$ and $\mathbf{w}_t^a = \mathbf{D}^T \boldsymbol{\Sigma}_a^{-1}(\boldsymbol{\mu}_a - \mathbf{a}_t)$.

For simplicity in our experiments we defined the first order time derivatives as:

$$\mathbf{d}_t = \mathbf{c}_{t+2} - \mathbf{c}_{t-2} \tag{B.9}$$

and the second order time derivatives as

$$\mathbf{a}_t = \mathbf{d}_{t+2} - \mathbf{d}_{t-2} = \mathbf{c}_{t+4} - \mathbf{c}_{t-4} - 2\mathbf{c}_t \ . \tag{B.10}$$

Then it can be verified that

$$\begin{aligned}\frac{\partial \mathbf{x}_t}{\partial \mathbf{H}} \mathbf{\Sigma}^{-1}(\boldsymbol{\mu} - \mathbf{x}_t) = (\mathbf{w}_t^c ./ \mathbf{y}_t)\mathbf{z}_t^T + (\mathbf{w}_t^d ./ \mathbf{y}_{t+2})\mathbf{z}_{t+2}^T - (\mathbf{w}_t^d ./ \mathbf{y}_{t-2})\mathbf{z}_{t-2}^T + \\ (\mathbf{w}_t^a ./ \mathbf{y}_{t+4})\mathbf{z}_{t+4}^T - (\mathbf{w}_t^a ./ \mathbf{y}_{t-4})\mathbf{z}_{t-4}^T - 2(\mathbf{w}_t^a ./ \mathbf{y}_t)\mathbf{z}_t^T \ .\end{aligned} \tag{B.11}$$

B.2 Gradient of MDE Performance Function

In this section we want to find an expression for $\frac{\partial J(\mathbf{\Lambda})}{\partial \mathbf{\Lambda}}$. We can start with

$$\frac{\partial J(\mathbf{\Lambda})}{\partial \mathbf{\Lambda}} = \sum_{k,j} S_{kj} \frac{\partial P(L_{kj}|\mathbf{X}_k, \mathbf{\Lambda})}{\partial \mathbf{\Lambda}} \ . \tag{B.12}$$

The posterior $P(L_{kj}|\mathbf{X}_k, \mathbf{\Lambda})$ can be expressed in terms of likelihoods applying Bayes' theorem:

$$P(L_{kj}|\mathbf{X}_k, \mathbf{\Lambda}) = \frac{p(\mathbf{X}_k|L_{kj}, \mathbf{\Lambda})P(L_{kj})}{\sum_u p(\mathbf{X}_k|L_{ku}, \mathbf{\Lambda})P(L_{ku})} \ . \tag{B.13}$$

The gradient of the posterior can then be computed as follows:

$$\begin{aligned}\frac{\partial P(L_{kj}|\mathbf{X}_k, \mathbf{\Lambda})}{\partial \mathbf{\Lambda}} = & \frac{P(L_{kj})}{\sum_u p(\mathbf{X}_k|L_{ku}, \mathbf{\Lambda})P(L_{ku})} \frac{\partial p(\mathbf{X}_k|L_{kj}, \mathbf{\Lambda})}{\partial \mathbf{\Lambda}} \\ & - \frac{p(\mathbf{X}_k|L_{kj}, \mathbf{\Lambda})P(L_{kj})}{\Big(\sum_u p(\mathbf{X}_k|L_{ku}, \mathbf{\Lambda})P(L_{ku})\Big)^2} \sum_u P(L_{ku}) \frac{\partial p(\mathbf{X}_k|L_{ku}, \mathbf{\Lambda})}{\partial \mathbf{\Lambda}} \ .\end{aligned} \tag{B.14}$$

Further, applying that

$$\frac{\partial p(\mathbf{X}_k|L_{kj}, \mathbf{\Lambda})}{\partial \mathbf{\Lambda}} = p(\mathbf{X}_k|L_{kj}, \mathbf{\Lambda}) \frac{\partial \log p(\mathbf{X}_k|L_{kj}, \mathbf{\Lambda})}{\partial \mathbf{\Lambda}} \tag{B.15}$$

we obtain the following expression:

$$\frac{\partial P(L_{kj}|\mathbf{X}_k,\mathbf{\Lambda})}{\partial \mathbf{\Lambda}} = P(L_{kj}|\mathbf{X}_k,\mathbf{\Lambda})\frac{\partial g_j(\mathbf{X}_k;\mathbf{\Lambda})}{\partial \mathbf{\Lambda}} - P(L_{kj}|\mathbf{X}_k,\mathbf{\Lambda})\sum_u P(L_{ku}|\mathbf{X}_k,\mathbf{\Lambda})\frac{\partial g_u(\mathbf{X}_k;\mathbf{\Lambda})}{\partial \mathbf{\Lambda}} \ , \quad \text{(B.16)}$$

where $g_j(\mathbf{X}_k;\mathbf{\Lambda}) = \log p(\mathbf{X}_k|L_{kj},\mathbf{\Lambda})$.

The gradient of the posterior given by Eq. B.16 can be substituted in Eq. B.12 to obtain two terms. The first term is given by

$$\sum_{k,j} S_{kj} P(L_{kj}|\mathbf{X}_k,\mathbf{\Lambda})\frac{\partial g_j(\mathbf{X}_k;\mathbf{\Lambda})}{\partial \mathbf{\Lambda}} \quad \text{(B.17)}$$

and the second term is given by

$$\begin{aligned} &- \sum_{k,j} S_{kj} P(L_{kj}|\mathbf{X}_k,\mathbf{\Lambda}) \sum_u P(L_{ku}|\mathbf{X}_k,\mathbf{\Lambda})\frac{\partial g_u(\mathbf{X}_k;\mathbf{\Lambda})}{\partial \mathbf{\Lambda}} \\ &= - \sum_k \Big(\sum_j S_{kj} P(L_{kj}|\mathbf{X}_k,\mathbf{\Lambda}) \Big) \sum_u P(L_{ku}|\mathbf{X}_k,\mathbf{\Lambda})\frac{\partial g_u(\mathbf{X}_k;\mathbf{\Lambda})}{\partial \mathbf{\Lambda}} \\ &= - \sum_k \bar{S}_k(\mathbf{\Lambda}) \sum_u P(L_{ku}|\mathbf{X}_k,\mathbf{\Lambda})\frac{\partial g_u(\mathbf{X}_k;\mathbf{\Lambda})}{\partial \mathbf{\Lambda}} \\ &= - \sum_k \bar{S}_k(\mathbf{\Lambda}) \sum_j P(L_{kj}|\mathbf{X}_k,\mathbf{\Lambda})\frac{\partial g_j(\mathbf{X}_k;\mathbf{\Lambda})}{\partial \mathbf{\Lambda}} \end{aligned} \quad \text{(B.18)}$$

Finally, combining Eqs. B.17 and B.18 results in

$$\frac{\partial J(\mathbf{\Lambda})}{\partial \mathbf{\Lambda}} = \sum_{k,j} (S_{kj} - \bar{S}_k) P(L_{kj}|\mathbf{X}_k,\mathbf{\Lambda})\frac{\partial g_j(\mathbf{X}_k;\mathbf{\Lambda})}{\partial \mathbf{\Lambda}} \ . \quad \text{(B.19)}$$

Bibliography

[1] C.-H. Lee, "From knowledge-ignorant to knowledge-rich modeling: a new speech research paradigm for next generation automatic speech recognition," in *Proc. ICSLP*, 2004.

[2] P. Ladefoged and K. Johnson, *A course in phonetics*, Wadsworth, 2010.

[3] B. Schuppler, *Automatic Analysis of Acoustic Reduction in Spontaneous Speech*, Ph.D. thesis, Radboud University Nijmegen, 2011.

[4] R. Maher, "Audio forensic examination," *IEEE Signal Processing Magazine*, vol. 26, no. 2, pp. 84 –94, march 2009.

[5] S. Frisch and R. Wright, "The phonetics of phonological speech errors: An acoustic analysis of slips of the tongue," *Journal of Phonetics*, vol. 30, no. 2, pp. 139–162, 2002.

[6] M. de Bruijn, L. ten Bosch, et al., "Objective acoustic-phonetic speech analysis in patients treated for oral or oropharyngeal cancer," *Folia Phoniatrica et Logopaedica*, vol. 61, pp. 180–187, 2009.

[7] A. Thambiratnam, *Acoustic keyword spotting in speech with applications to data mining*, Ph.D. thesis, Queensland University of Technology, 2005.

[8] S. Iseji T. Nitta et al., "Key-word spotting using phonetic distinctive features extracted from output of an LVCSR engine," in *Proc. SSPR*, 2003, paper MAP16.

[9] A. Jansen and P. Niyogi, "Point process models for spotting keywords in continuous speech," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 17, no. 8, pp. 1457–1470, nov. 2009.

[10] S. M. Witt, *Use of Speech Recognition in Computer-Assisted Language Learning*, Ph.D. thesis, Cambridge University, 1999.

[11] A. Neri, C. Cucchiarini, and W. Strik, "Automatic speech recognition for second language learning: How and why it actually works," in *Proc. ICPhS*, 2003, pp. 1157–1160.

[12] J. Xu, J. Xin Liu, et al., "Design of the pronunciation dictionary for an English CAPT system," in *Proc. ICCDA*, 2010, vol. 4, pp. V4–9 –V4–13.

[13] J. Li and C.-H. Lee, "On designing and evaluating speech event detectors," in *Proc. Interspeech*, 2005, pp. 3365–3368.

[14] I. Bromberg et al., "Detection-based ASR in the automatic speech attribute transcription project," in *Proc. ASRU*, 2007, pp. 1829–1832.

[15] C. Ma, *A detection-based pattern recognition framework and its applications*, Ph.D. thesis, Georgia Tech, April 2010.

[16] S. M. Siniscalchi and C.-H. Lee, "A study on integrating acoustic-phonetic information into lattice rescoring for automatic speech recognition," *Speech Communication*, vol. 51, no. 11, pp. 1139–1153, 2009.

[17] S. M. Siniscalchi, T. Svendsen, and C.-H. Lee, "Towards bottom-up continuous phone recognition," in *Proc. ASRU*, 2007, pp. 566 –569.

[18] S. M. Siniscalchi, T. Svendsen, and C.-H. Lee, "A bottom-up stepwise knowledge-integration approach to large vocabulary continuous speech recognition using weighted finite state machines," in *Proc. Interspeech*, 2011, pp. 901–904.

[19] H.-C. Wang C.-Y. Lin, "Attribute-based mandarin speech recognition using conditional random fields," in *Proc. Interspeech*, 2007, pp. 1833–1836.

[20] C. Zhang, Y. Liu, and C.-H. Lee, "Detection-based accented speech recognition using articulatory features," in *Proc. ASRU*, 2011, pp. 500–505.

[21] S. A. Liu, *Landmark detection for distinctive feature-based speech recognition.*, Ph.D. thesis, Dept. of Electrical Engineering and Computer Science (Massachusetts Institute of Technology), 1995.

[22] A. Juneja, *Speech recognition based on phonetic features and acoustic landmarks*, Ph.D. thesis, University of Maryland College Park, 2004.

[23] M. Hasegawa-Johnson, J. Baker, et al., "Landmark-based speech recognition: Report of the 2004 Johns Hopkins summer workshop," in *Proc. ICASSP*, 18-23 2005, vol. 1, pp. 213 – 216.

[24] A. Juneja and C. Y. Espy-Wilson, "A probabilistic framework for landmark detection based on phonetic features for automatic speech recognition," *Journal of the Acoustical Society of America*, pp. 1154–1168, June 2008.

[25] L. Bahl, Brown, et al., "Maximum mutual information estimation of hidden markov model parameters for speech recognition," in *Proc. ICASSP*, apr 1986, vol. 11, pp. 49–52.

[26] D. Povey, *Discriminative Training for Large Vocabulary Speech Recognition*, Ph.D. thesis, Cambridge University Engineering Dept, 2003.

[27] B.-H. Juang, W. Hou, and C.-H. Lee, "Minimum classification error rate methods for speech recognition," *IEEE Trans. Speech and Audio Processing*, vol. 5, no. 3, pp. 257–265, 1997.

[28] X. He, L. Deng, and W. Chou, "Discriminative learning in sequential pattern recognition," *IEEE Signal Processing Magazine*, vol. 25, no. 5, pp. 14 –36, 2008.

[29] Q. Fu, *A generalization of the minimum classification error (MCE) training method for speech recognition and detection*, Ph.D. thesis, Georgia Institute of Technology, 2008.

[30] A. Biem, S. Katagiri, et al., "An application of discriminative feature extraction to filter-bank-based speech recognition," *IEEE Trans. Speech and Audio Processing*, vol. 9, no. 2, pp. 96–110, feb 2001.

[31] A. Biem, "Optimizing features and models using the minimum classification error criterion," in *Proc. ICASSP*, 2003.

[32] B. Mak, Y.-C. Tam, and R. Hsiao, "Discriminative training of auditory filters of different shapes for robust speech recognition," in *Proc. ICASSP*, 2003, vol. 2, pp. II – 45–8.

[33] H. Bořil, P. Fousek, and P. Pollák, "Data-driven design of front-end filter bank for Lombard speech recognition," in *Proc. ICSLP*, Pittsburgh, Pennsylvania, 2006, pp. 381 – 384.

[34] H. Huang and J. Zhu, "Minimum phoneme error based filter bank analysis for speech recognition," in *Proc. Multimedia and Expo*, 2006, pp. 1081 –1084.

[35] L. Burget and H. Hermansky, "Data driven design of filter bank for speech recognition," in *Proc. ICTSD*, 2001, pp. 299–304.

[36] Y. Suh and H. Kim, "Data-driven filter-bank-based feature extraction for speech recognition," in *Proc. SPECOM*, 2004, p. 154.

[37] C. Lee, D. Hyun, et al., "Optimizing feature extraction for speech recognition," *IEEE Trans. Speech and Audio Processing*, vol. 11, no. 1, pp. 80–87, Jan. 2003.

[38] Y. Bengio, R. De Mori, et al., "Global optimization of a neural network-hidden Markov model hybrid," *IEEE Trans. Neural Networks*, vol. 3, no. 2, pp. 252 –259, mar 1992.

[39] F. T. Johansen, "Global optimisation of HMM input transformations," in *Proc. ICSLP*, 1994, pp. 239–242.

[40] B. Zamani, A. Akbari, et al., "Optimized discriminative transformations for speech features based on minimum classification error," *Pattern Recognition Letters*, vol. 32, no. 7, pp. 948 – 955, 2011.

[41] J. Droppo and A. Acero, "Joint discriminative front end and back end training for improved speech recognition accuracy," in *Proc. ICASSP*, May 2006, vol. 1, p. I.

[42] M. M. Doss F. Valente and W. Wang, "Analysis and comparison of recent MLP features for LVCSR systems," in *Proc. Interspeech*, 2011, pp. 1245–1248.

[43] D. Povey, B. Kingsbury, et al., "fMPE: Discriminatively Trained Features for Speech Recognition," in *Proc. ICASSP*, 2005, vol. 1, pp. 961 – 964.

[44] B. Zhang, S. Matsoukas, and R. Schwartz, "Discriminatively trained region dependent feature transforms for speech recognition," in *Proc. ICASSP*, May 2006, vol. 1, p. I.

[45] J. Zheng, O. Cetin, et al., "Combining discriminative feature, transform, and model training for large vocabulary speech recognition," in *Proc. ICASSP*, 2007, vol. 4, pp. 633–636.

[46] R. Hsiao and B. Mak, "Discriminative feature transformation by guided discriminative training," in *Proc. ICASSP*, May 2004, vol. 1, pp. 897–900.

[47] A. M. Canterla and M. H. Johnsen, "Optimized Feature Extraction and HMMs in Subword Detectors," in *Proc. Interspeech*, 2011, pp. 2397–2400.

[48] A. M. Canterla and M. H. Johnsen, "Minimum Detection Error Training of Subword Detectors," in *Proc. ASRU*, 2011, pp. 506–5011.

[49] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, Wiley, 2nd edition, 2001.

[50] K. Fukunaga, *Introduction to statistical pattern recognition*, Academic Press Professional, Inc., 2nd edition, 1990.

[51] C.-H. Lee and Q. Huo, "On adaptive decision rules and decision parameter adaptation for automatic speech recognition," in *Proc. IEEE*, 2000, pp. 1241–1269.

[52] H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," *Journal of the Acoustical Society of America*, vol. 87, no. 4, pp. 1738–1752, 1990.

[53] X. Huang, A. Acero, and H.-W. Hon, *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*, Prentice Hall PTR, 2001.

[54] R. Schwartz and Y.-L. Chow, "The N-best algorithms: an efficient and exact procedure for finding the N most likely sentence hypotheses," in *Proc. ICASSP*, Apr. 1990, pp. 81 –84 vol.1.

[55] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. 28, no. 4, pp. 357 – 366, Aug. 1980.

[56] N. Malayath and H. Hermansky, "Data-driven spectral basis functions for automatic speech recognition," *Speech Communication*, vol. 40, no. 4, pp. 449–466, 2003.

[57] S. Furui, "Speaker-independent isolated word recognition using dynamic features of speech spectrum," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. 34, no. 1, pp. 52 – 59, feb 1986.

[58] J. Frankel, M Wester, and S. King, "Articulatory feature recognition using dynamic Bayesian networks," *Computer Speech and Language*, vol. 21, no. 4, pp. 620–640, October 2007.

[59] C. Ma, Y. Tsao, and C.-H. Lee, "A study on detection based automatic speech recognition," in *Proc. Interspeech*, 2006.

[60] C.-H. Lee, "A tutorial on speaker and speech verification," in *Proc. NORSIG*, 1998, number 9-16.

[61] S. G. Pettersen, M. H. Johnsen, and T. A. Myrvoll, "Task independent speech verification using SB-MVE trained phone models," in *Proc. Robust2004*, 2004, number 10.

[62] M. Sokolova, N. Japkowicz, and S. Szpakowicz, "Beyond Accuracy, F-Score and ROC: A Family of Discriminant Measures for Performance Evaluation," *AI 2006: Advances in Artificial Intelligence*, pp. 1015–1021, 2006.

[63] A. Martin, G. Doddington, et al., "The det curve in assessment of detection task performance," in *Proc. Eurospeech*, 1997, pp. 1895–1898.

[64] Carla Lopes and Fernando Perdigo, "Improved performance evaluation of speech event detectors," in *Proc. Interspeech*, 2006.

[65] Ø. Birkenes, *A Framework for Speech Recognition using Logistic Regression*, Ph.D. thesis, Norwegian University of Science and Technology, Faculty of Information Technology, Mathematics and Electrical Engineering, 2007.

[66] A. Y. Ng and M. I. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," in *Proc. NIPS*, 2001, number 14.

[67] G. Bouchard and B. Triggs, "The trade-off between generative and discriminative classifiers," in *Proc. Computational Statistics*, 2004, pp. 721–728.

[68] P. Liang and M. I. Jordan, "An asymptotic analysis of generative, discriminative, and pseudolikelihood estimators," in *Proc. ICML*, 2008, pp. 584–591.

[69] Ralf Schlüter, Wolfgang Macherey, et al., "Comparison of discriminative training criteria and optimization methods for speech recognition," *Speech Communication*, vol. 34, no. 3, pp. 287 – 310, 2001.

[70] X. He and L. Deng, *Discriminative Learning for Speech Recognition: Theory and Practice*, Synthesis Lectures on Speech and Audio Processing. Morgan & Claypool Publishers, 2008.

[71] J. Keshet, C.-C. Cheng, et al., "Direct error rate minimization of hidden markov models," in *Proc. Interspeech*, 2011, pp. 449–452.

[72] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," in *Proc. Neural Networks*, 1993, pp. 586–591.

[73] E. McDermott, *Discriminative Training for Speech Recognition*, Ph.D. thesis, Waseda University, 1997.

[74] E. McDermott, T. Hazen, et al., "Discriminative training for large-vocabulary speech recognition using minimum classification error," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 15, no. 1, pp. 203 –223, 2007.

[75] M. Gibson and T. Hain, "Error approximation and minimum phone error acoustic model estimation," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 18, pp. 1269–1279, August 2010.

[76] B. Launay, O. Siohan, et al., "Towards knowledge-based features for HMM based large vocabulary automatic speech recognition," in *Proc. ICASSP*, 2002, vol. 1, pp. I–817 – I–820 vol.1.

[77] P. Niyogi and M. M. Sondhi, "Detecting stop consonants in continuous speech," vol. 111, no. 2, pp. 1063–1076, Feb. 2002.

[78] K.K. Paliwal, "Lexicon-building methods for an acoustic sub-word based speech recognizer," in *Proc. ICASSP*, Apr. 1990, pp. 729 –732 vol.2.

[79] M. Ostendorf, "Moving beyond the 'beads-on-a-string' model of speech," in *Proc. ASRU*, 1999, pp. 79–84.

[80] K. Livescu, O. Çetin, et al., "Articulatory feature-based methods for acoustic and audio-visual speech recognition: Summary from the 2006 JHU Summer Workshop," in *Proc. ICASSP*, Honolulu, April 2007.

[81] I-F. Chen and H.-M. Wang, "An investigation of phonological feature systems used in detection-based ASR," in *Proc. ISCSLP*, 2008.

[82] N. Chomsky and M. Halle, *The Sound Pattern of English*, Harper & Row, New York, 1968.

[83] S. King and P. Taylor, "Detection of phonological features in continuous speech using neural networks," *Computer Speech and Language*, vol. 14(4), pp. 333–353, 2000.

[84] I-F. Chen and H.-M. Wang, "Articulatory feature asynchrony analysis and compensation in detection-based ASR," in *Proc. Interspeech*, 2009.

[85] K. Erier and G. Freeman, "Using articulatory features for speech recognition," in *Proc. Communications, Computers, and Signal Processing*, may 1995, pp. 562–566.

[86] S. M. Siniscalchi, Ø. Birkenes, et al., "Joint optimization of event detectors and evidence merger for continuous phone recognition," in *Proc. SPKD*, 2008.

[87] S. M. Siniscalchi, T. Svendsen, and C.-H. Lee, "A penalized logistic regression approach to detection based phone classification," in *Proc. Interspeech*, 2008, pp. 2390–2393.

[88] H. Bourlard and N. Morgan, "Continuous speech recognition by connectionist statistical methods," *IEEE Trans. Neural Networks*, vol. 4, no. 6, pp. 893 –909, Nov. 1993.

[89] J. Morris and E. Fosler-Lussier, "Conditional random fields for integrating local discriminative classifiers," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 16, no. 3, pp. 617–628, March 2008.

[90] J. S. Garofolo et al., "DARPA TIMIT acoustic phonetic continuous speech corpus CDROM," 1993.

[91] K.-F. Lee and H.-W. Hon, "Speaker-independent phone recognition using hidden Markov models," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. 37, no. 11, pp. 1641–1648, 1989.

[92] S. Young, D. Kershaw, et al., *The HTK Book Version 3.4*, Cambridge University Press, 2006.

[93] E. McDermott and S. Katagiri, "String-level MCE for continuous phoneme recognition," in *Proc. Eurospeech*, 1997, pp. 123–126.

[94] S. A. Schane, *Generative phonology*, Prentice-Hall Englewood Cliffs, N.J., 1973.

[95] V. Fromkin and R. Rodman, *An Introduction to Language, 5th Ed.*, Harcourt Brace College Publishers, Fort Worth, TX, 1993.